SUNG KYUN KWAN UNIVERSITY

설계 명세서

Piggy-b

낙전 방지 동전 적립 시스템

SUNG KYUN KWAN UNIVERSITY

목차

1. Preface

1.1 Objectives

Preface 에서는 본 문서의 대상 독자들과, 문서의 전반적인 구조 그리고 문서의 각 부분의 역할에 대해 설명한다. 그리고 버전 관리 정책과 버전 변경 기록, 문서의 변경 사항들을 서술한다.

1.2 Readership

본 설계 명세서는 목표 시스템의 개발 및 유지 보수를 담당하는 모든 사람들을 대상 독자로 한다. 본 문서의 독자로 선정된 사람들로는 실제로 개발을 담당하는 소프트웨어 엔지니어(software engineers)부터, 시스템을 설계하는 시스템 아키텍처(system architecture)와 유지보수를 담당하는 모든 외주 업체들 등이 있다.

1.3 Document Structure

이 문서는 총 14개의 장으로 구성되어 있다. 14개의 장은 Preface, Introduction, System Architecture, Common Components, User Activity, 사용자 관리 Subsystem, 동전 관리 Subsystem, 환급 Subsystem, Communication Protocol, Database System, Testing Plan, Development Environment, Index, Reference로 구성되어 있다. 각 장의 역할과 내용은 다음과 같다.

A Preface

Preface에서는 이 문서의 대상 독자들과 문서의 전체 구조, 각 부문의 내용에 대해 설명한다. 그리고 버전 관리 정책, 버전 변경 기록, 문서의 변경 사항들을 서술한다.

B Introduction

Introduction에서는 사용자 관점에서의 시스템에 대해 서술한다.

C System Architecture

System Architecture에서는 전체 시스템에 대한 개요를 서술한다. 시스템의 구조를 Block diagram으로 설명하고, 컴포넌트의 관계를 Package diagram, Deployment diagram으로 표현한다. 각 서브 시스템의 자세한 설명은 단원을 나누어 설명한다.

D Common Components

Common Components에는 모든 서브시스템에서 공용으로 사용하는 컴포넌트에 대해 설명한다.

E User Activity

Piggy-B 시스템에서 사용자가 사용할 수 있는 인터페이스에 관한 기능을 Class diagram, Sequence diagram, State diagram을 통해 서술한다.

F 사용자 관리 Subsystem

서브시스템 중 하나인 사용자 관리 서브시스템의 설계를 Class diagram, Sequence Diagram, State Diagram을 통해 서술한다.

G 동전 관리 Subsystem

서브시스템 중 하나인 동전 관리 서브시스템의 설계를 Class diagram, Sequence

Diagram, State Diagram을 통해 서술한다.

H 환급 Subsystem

서브시스템 중 하나인 환급 관리 서브시스템의 설계를 Class diagram, Sequence Diagram, State Diagram을 통해 서술한다.

I Communication Protocol

Communication Protocol에서는 서버와 안드로이드 스마트폰이 통신할 때 사용하는 프로토콜에 대해서 서술한다. 클라이언트-서버 모델에서는 서버는 별도의 클라이언트 요청 없이 클라이언트로 요청을 보낼 수 없으므로, 모든 프로토콜은 안드로이드에서 서버로 전송되는 요청을 표현한다.

J Database Design

Database Design에서는 서버 내 저장되는 데이터의 구조 및 저장 방법을 서술한다. 요구사항에 기반한 데이터베이스의 ER diagram을 제시하고, 이에 대한 relational schema를 서술한다. Normalization 과정을 통하여 데이터베이스 간에 존재할 수 있는 중복을 방지하고, SQL DDL을 작성한다.

K Testing Plan

Testing Plan에서는 테스트 정책과 테스트 케이스에 대해 설명한다.

L Development Environment

Development Environment에서는 개발 환경, 언어, 버전 관리 도구, 실행 환경 등에 대해 설명한다.

M Index

Index에서는 문서를 빠르게 색인 할 수 있도록 인덱스를 제공한다. 다이어그램 및 표 등의 인덱스를 제공한다.

1.4 Version of the Document

A Version Format

버전 번호는 major number와 minor number로 구성되어 있으며, "(major number).(minor number)"의 형태로 표현되고, 본 문서의 버전은 0.1부터 시작한다.

B Version Management Policy

본 설계 명세서가 수정될 때 마다 버전을 업데이트 한다. 하지만 변경 사이의 간격이 1시간 이내일 때에는 버전 번호를 업데이트 하지 않고 하나의 업데이트로 간주한다. 이미 완성된 부분을 변경할 때에는 minor number를 변경하며, 새로운 부분을 추가하거나 문서 구성이 예전에 비해 괄목할만한 변화가 있을 경우에는 major number를 변경한다.

C Version Update History

Version	Modified Date	Explanation
0.1	2016-05-29	문서의 초안과 표지 작성
1.0	2016-05-29	Preface, Introduction 작성
1.1	2016-05-29	문서 구성 수정
2.0	2016-05-30	System Architecture 작성

3.0	2016-05-31	사용자 관리 Subsystem 작성
3.1	2016-06-01	사용자 관리 Subsystem Diagram 수정
4.0	2016-06-02	동전 관리 Subsystem 작성
4.1	2016-06-02	동전 관리 Subsystem 오타 수정
4.2	2016-06-02	동전 관리 Subsystem Diagram 수정
5.0	2016-06-03	환급 Subsystem 작성
6.0	2016-06-03	Common Components, Communication Protocol, Database Design 작성
6.1	2016-06-03	Communication Protocol 수정
7.0	2016-06-04	Testing Plan, Development Environment 작성
7.1	2016-06-04	Testing Plan 순서 수정
7.2	2016-06-04	Development Environment 내용 수정
7.3	2016-06-04	Development Environment 사진 수정
8.0	2016-06-05	Index 작성, 문서 전체 오타 수정

2. Introduction

2.1 Objective

이 장에서는 본 프로젝트의 범위와 소프트웨어의 환경, 그리고 그때 발생하는 주요 제약사항들에 대해 설명한다. 또한, 시스템의 설계에 사용한 UML(Unified Modeling Language)의 다양한 다이어그램 작성을 위해 사용한 개발 툴(tool)을 소개한다.

2.2 Project Scope

Piggy-B 시스템은 사용자가 Piggy-B 가맹점에서 물건을 현금으로 구입 후 발생하는 잔돈을 직접 받지 않고 디지털 코인의 형태로 사용자의 Piggy-B 계정에 적립하는 시스템이다. Piggy-B 시스템은 크게 세개의 하드웨어와 세가지 서브시스템으로 나눌 수 있다.

먼저, 하드웨어는 POS 기, NFC 리더기, 그리고 모바일 디바이스이다. POS 기는 가맹점 용 Piggy-B 프로그램이 설치될 컴퓨터이다. NFC 는 가맹점에서 구비해야 할 하드웨어로 NFC 통신을 위해 필요하다. 모바일 디바이스는 사용자가 해당 서비스를 이용하기 위해서 구매 시 가지고 있어야한다.

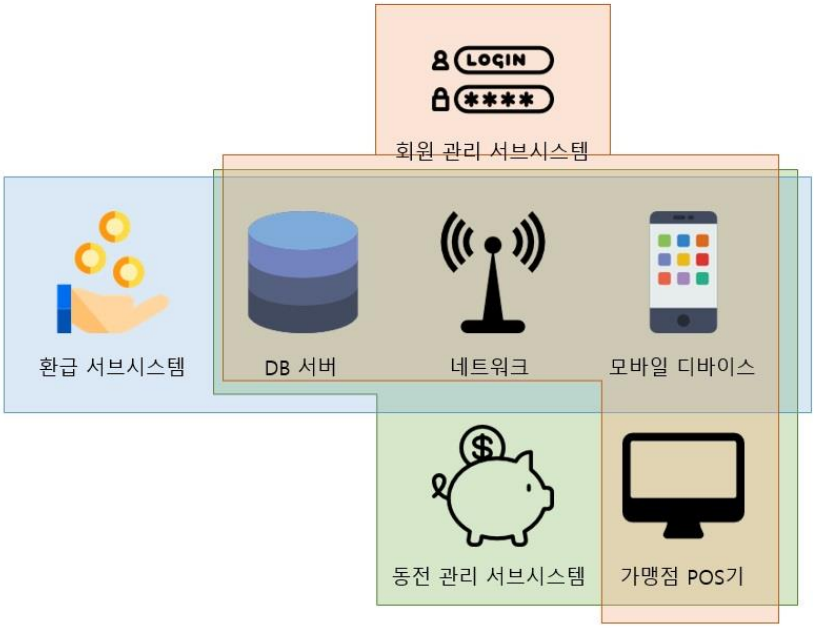

Figure 1 Piggy-B 시스템의 전체 구조

서브시스템은 크게 사용자 관리 서브시스템, 동전 관리 서브시스템, 환급 서브시스템으로 나눌 수 있다. 이 세가지 서브시스템은 Wi-Fi 등 네트워크에 연결된 서버의 통제를 받으며, 사용자의 모바일 디바이스와 가맹점의 POS 기는 네트워크에 연결되어 있어야 한다. 중앙 서버는 모든 서브시스템과 연계되어 있다. 사용자 관리 서브시스템에서는 사용자 모바일 디바이스로부터의 로그인 요청이나 회원가입 요청이 있을 때, 서버와 통신해 데이터를 대조, 저장하게 된다. 동전 관리 서브시스템은 먼저 NFC 를 통해 POS 기에서 사용자 모바일 디바이스로 전송하고, 사용자의 모바일 디바이스는 네트워크에 연결된 상태에서 서버와 연결해 NFC 데이터를 서버에 전송하며 서버는 전송 된 데이터를 저장한다. 환급 서브시스템에서는 환급 요청 발생 시 서버에서 은행에 환급 요청을 하게 된다.

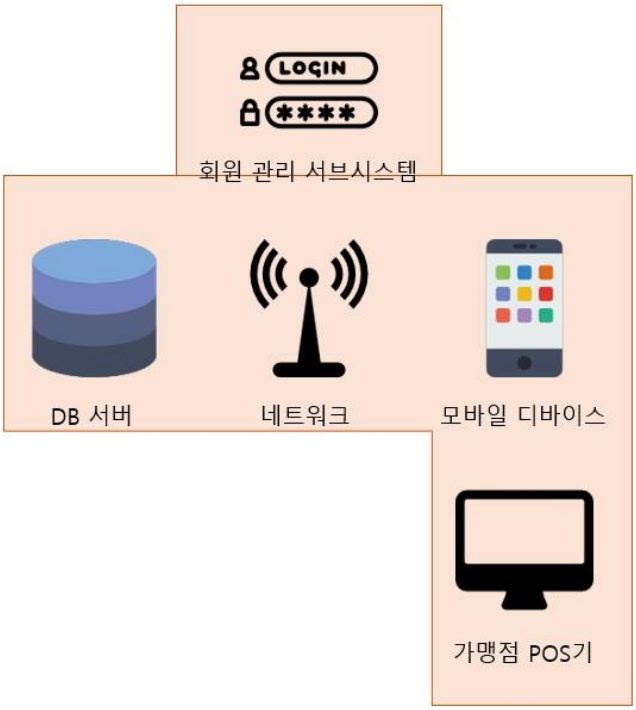

Figure 2 사용자 관리 시스템의 구조

사용자 관리 서브시스템은 고객 관리 서브시스템과 가맹점 관리 서브시스템으로 다시 나눌 수 있다. 고객 관리 서브시스템은 Piggy-B 사용자의 회원가입 시 입력해야하는 아이디, 비밀번호, 이름, 계좌 정보 등 고객 정보를 저장하며 로그인 시 입력된 값과 저장된 값이 일치하는지 대조한다. 가맹점 관리 서브시스템은 Piggy-B 가맹점의 정보를 저장한다.

Figure 3 동전 관리 시스템의 구조

다음으로 동전 관리 서브시스템은 동전 적립과 관리를 담당한다. 사용자가 가맹점에서 거래 후 사용자의 모바일 디바이스를 NFC 리더기에 태그 하면 가맹점의 POS 기에서 사용자 모바일 디바이스로 NFC 를 통해 적립 금액, 매장 정보 등 데이터가 전송된다. 동전 관리 서브시스템에서는 이러한 새로 전달된 NFC 데이터를 서버에 저장, 갱신한다.

Figure 4 환급 시스템의 구조

환급 서브시스템은 사용자의 환급 요청이 발생하면 서버는 은행에 환급을 요청하고, 사용자의 적립금이 기준 환급 금액 이상인지 확인한다. 환급 과정이 성공적으로 이루어졌을 때 사용자에게 성공을 알린다.

2.3 Major Constraints

A Deployment Issues

Piggy-B 시스템은 Piggy-B 서비스와 가맹을 맺은 상점에 한해서만 POS 기에 Piggy-B 프로그램을 설치하고 사용할 수 있다. Piggy-B 사용자들도 가맹점에서 거래를 한 경우에 한해서만 서비스를 이용할 수 있다. 따라서, Piggy-B 서비스가 상용화되기 위해서 해당 서비스와 가맹을 맺은 상점 수가 많아야 한다. Piggy-B 개발 후 해당 서비스를 사용할 가맹점을 모집하는 과정이 필수적이다.

B Transferring Constraints

Piggy-B 사용자는 해당 서비스 사용 중에 적립한 금액을 계좌를 통해 환급 받을 수 있다. 사용자가 환급을 요청하면 Piggy-B 서비스는 은행을 통해 사용자에게 적립금을 환급해주는데, 이때 수수료가 발생하게 된다. 무분별한 환급 요청으로 발생하는 수수료를 방지하기 위해서 환급 기준 금액을 설정했다. 그 이후에 발생할 수 밖에 없는 수수료에 대해서는 은행과 추후 계약을 맺어 수수료를 최소화 시켜야 한다.

2.4 Applied Tool

Figure 5 Gliffy

본 문서에 첨부되어 있는 모든 시스템 설계도와 diagram 의 작성에는 “Gliffy Diagrams” tool 을 사용하였다. “Gliffy Diagrams”는 구글 크롬 브라우저의 앱 스토어에서 무료로 설치할 수 있는 소프트웨어로 다양한 diagram 및 flow chart 를 그리는 데에 최적화 되어있는 소프트웨어이다. 특히, 이 소프트웨어는 온라인 상에서 작업이 가능하며 완성한 diagram 을 PNG, JPEG 등 다양한 형식의 확장자로 저장이 가능하다.

3. System Architecture

3.1 Objectives

System Architecture 에서는 해당 시스템의 architecture 에 대한 high-level 수준의 개요와 시스템 기능의 전체적 분포를 보여준다. 전체 시스템의 system organization 은 block diagram 으로 도식화 하였고, subsystem 들의 관계와 해당 시스템의 실제 배포 형태를 package diagram 및 deployment diagram 을 사용해 도식화 했다.

3.2 System Organization

A Overview

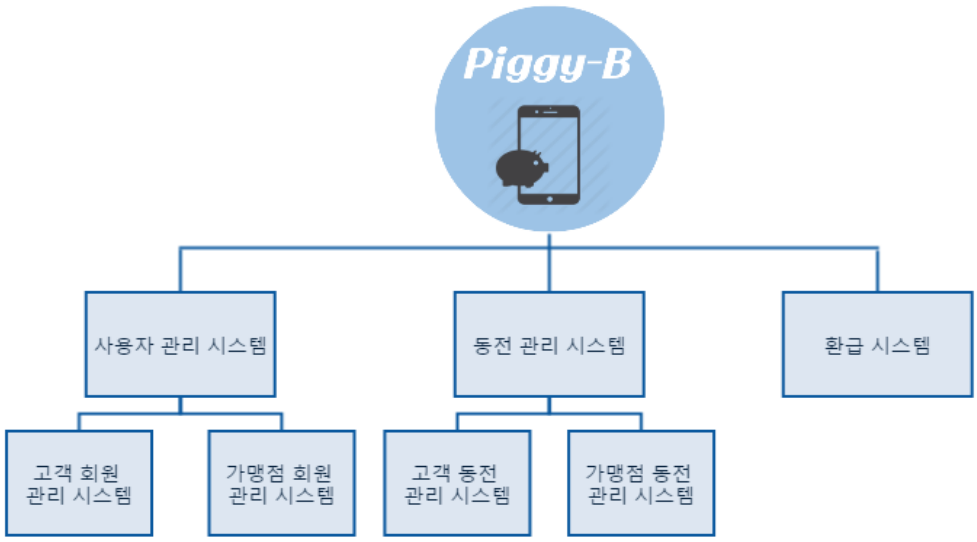

Diagram 1 전체 시스템 아키텍쳐 오버뷰

A.1 Application Architecture

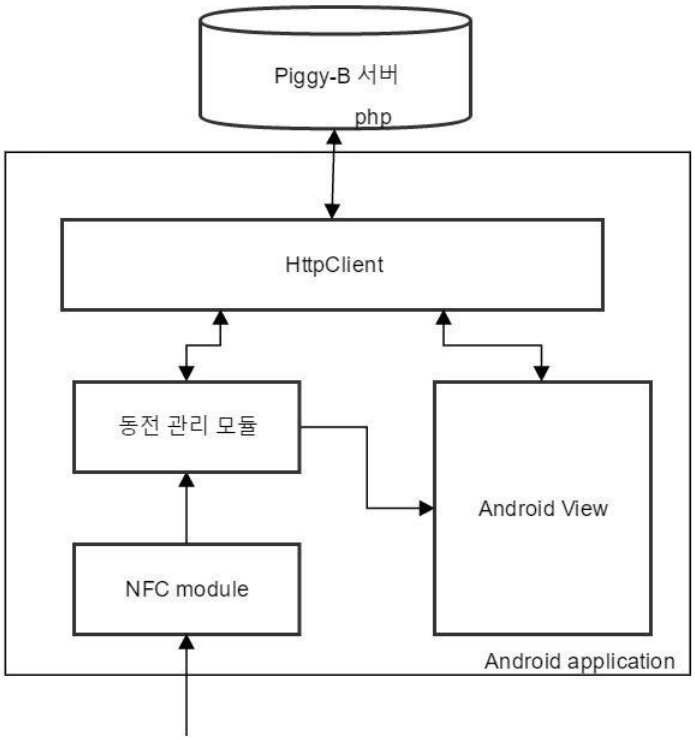

Diagram 2 Piggy-B시스템에서 안드로이드 클라이언트 어플리케이션의 전체 시스템 아키텍쳐

Piggy-B 어플리케이션은 동전 관리 모듈, NFC 모듈, HTTP 통신을 위한 HttpClient 모듈로 구성되어 있다. NFC trigger event 로 사용자가 적립할 동전에 대한 데이터의 송수신이 일어난다. 수신된 NFC 신호로부터 NFC 모듈은 유용한 정보로 변환하여 각각 동전 적립 모듈과 동전 관리 모듈로 전송된다. 동전 관리 모듈은 적립된 동전의 정보를 보관하고, 적립된 동전의 내역을 관리한다. HttpClient 모듈은 Piggy-B 서버에 있는 php 파일을 이용하여 Piggy-B 서버로 사용자의 정보를 요청하거나, 전송하는 통신 모듈이다. Android view 를 통해서 사용자가 요청한 정보를 보여준다.

A.2 Piggy-B Server Architecture

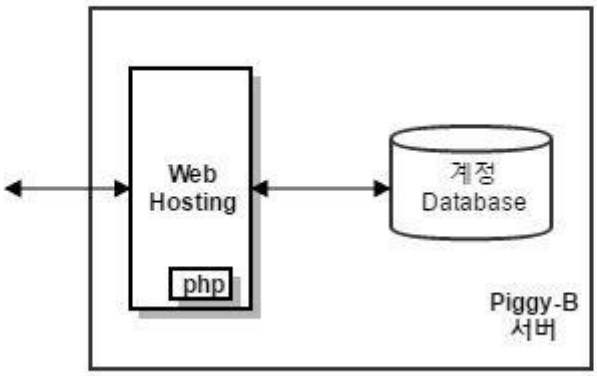

Diagram 3 Piggy-B 서버 아키텍처

안드로이드는 DB 서버에 직접 접속할 수 없기 때문에 HTTP 프로토콜을 기반으로 웹 호스팅 시스템을 사용해 Piggy-B 데이터베이스를 구축한다. 웹 호스팅 시스템에 PHP 파일을 읽어서 데이터베이스의 정보를 읽어오고, 저장한다.

B 사용자 관리 Subsystem

B.1 고객 회원 관리

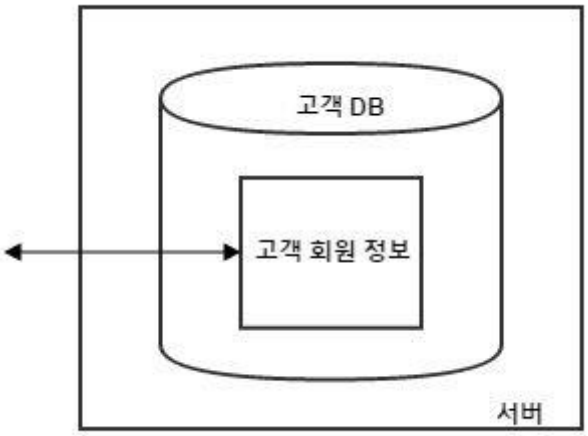

Diagram 4 고객 회원 정보 관리

Piggy-B 서비스를 이용하기 위해서는 회원가입 절차가 필수적이며, 회원가입 시 입력된 사용자의 정보가 서버의 DB 에 저장된다. 고객의 로그인 요청이나 회원정보 수정 요청 시 고객의 정보를 조회하고 고객에게 전달한다.

B.2 가맹점 회원 관리

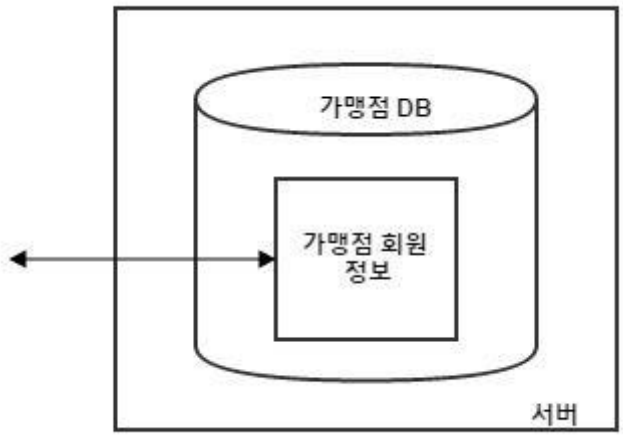

Diagram 5 가맹점 회원 정보 관리

Piggy-B 의 가맹점으로 등록된 상점의 정보가 서버의 DB 에 저장된다. 가맹점이 로그인 또는 회원정보 수정 요청 시 서버는 가맹점의 정보를 조회, 전달한다.

C 동전 관리 Subsystem

C.1 고객 동전 관리

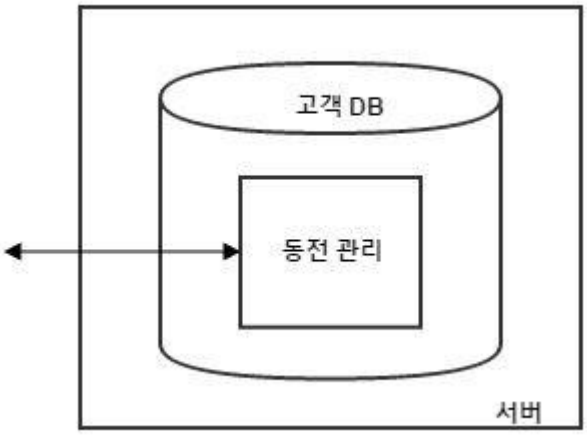

Diagram 6 고객 동전 관리

Piggy-B 서비스를 이용한 고객들의 사용 내역을 저장한다. 현재 총 적립금과 적립한 금액과 환급한 금액에 관한 이력을 저장한다.

C.2 가맹점 동전 관리

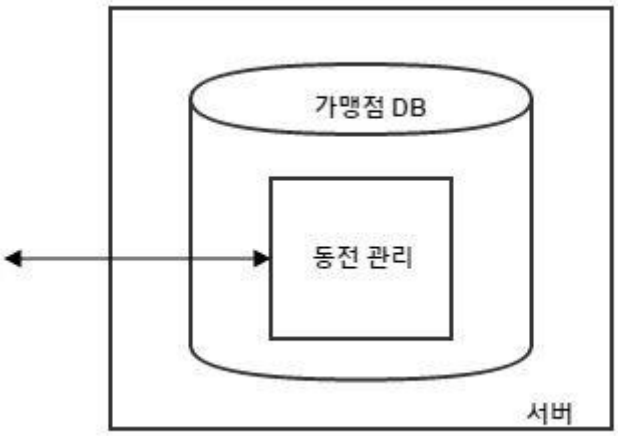

Diagram 7 가맹점 동전 관리

Piggy-B 의 가맹점이 가지고 있는 데이터를 저장한다. 각 가맹점에서 발생한 여러 Piggy-B 서비스 사용자들의 내역을 보관한다.

D 환급 Subsystem

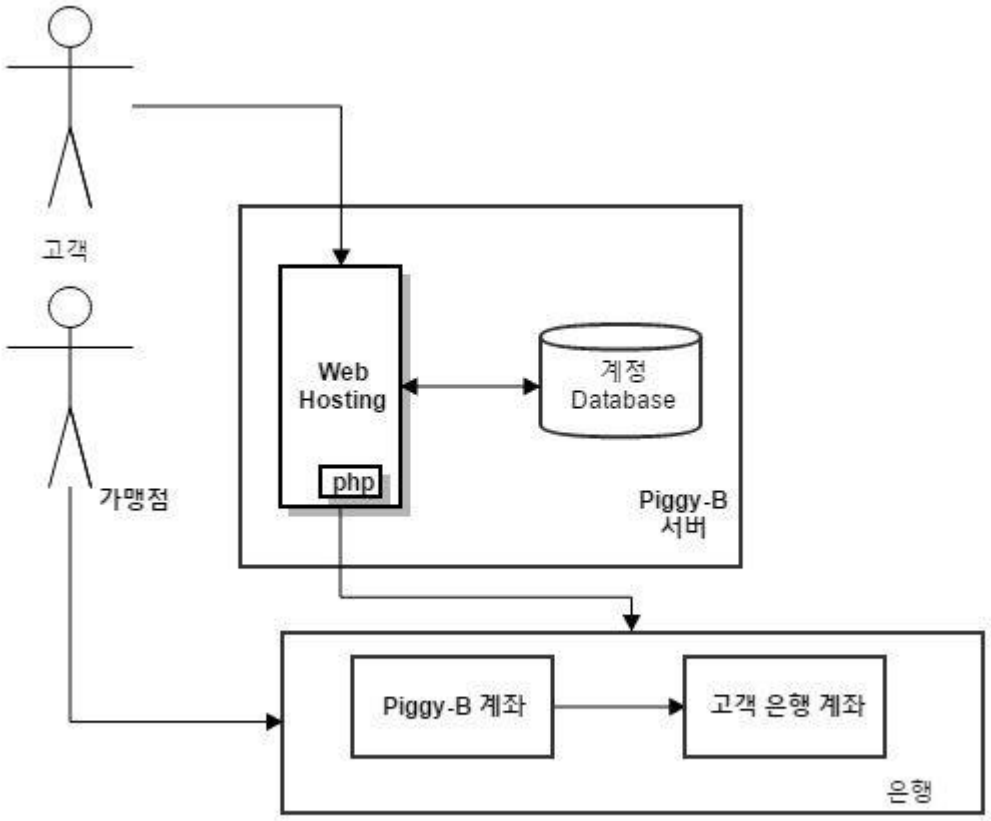

Diagram 8 환급 서브시스템

Piggy-B 서비스에 가입한 가맹점은 Piggy-B 은행 계좌에 적립된 금액을 송금한다. 고객이 Piggy-B 에 환급 요청을 하면 Piggy-B 시스템은 Piggy-B 계좌에서 환급을 요청한 고객의 은행 계좌로 이체할 것을 은행에 요청한다.

3.3 Package Diagram

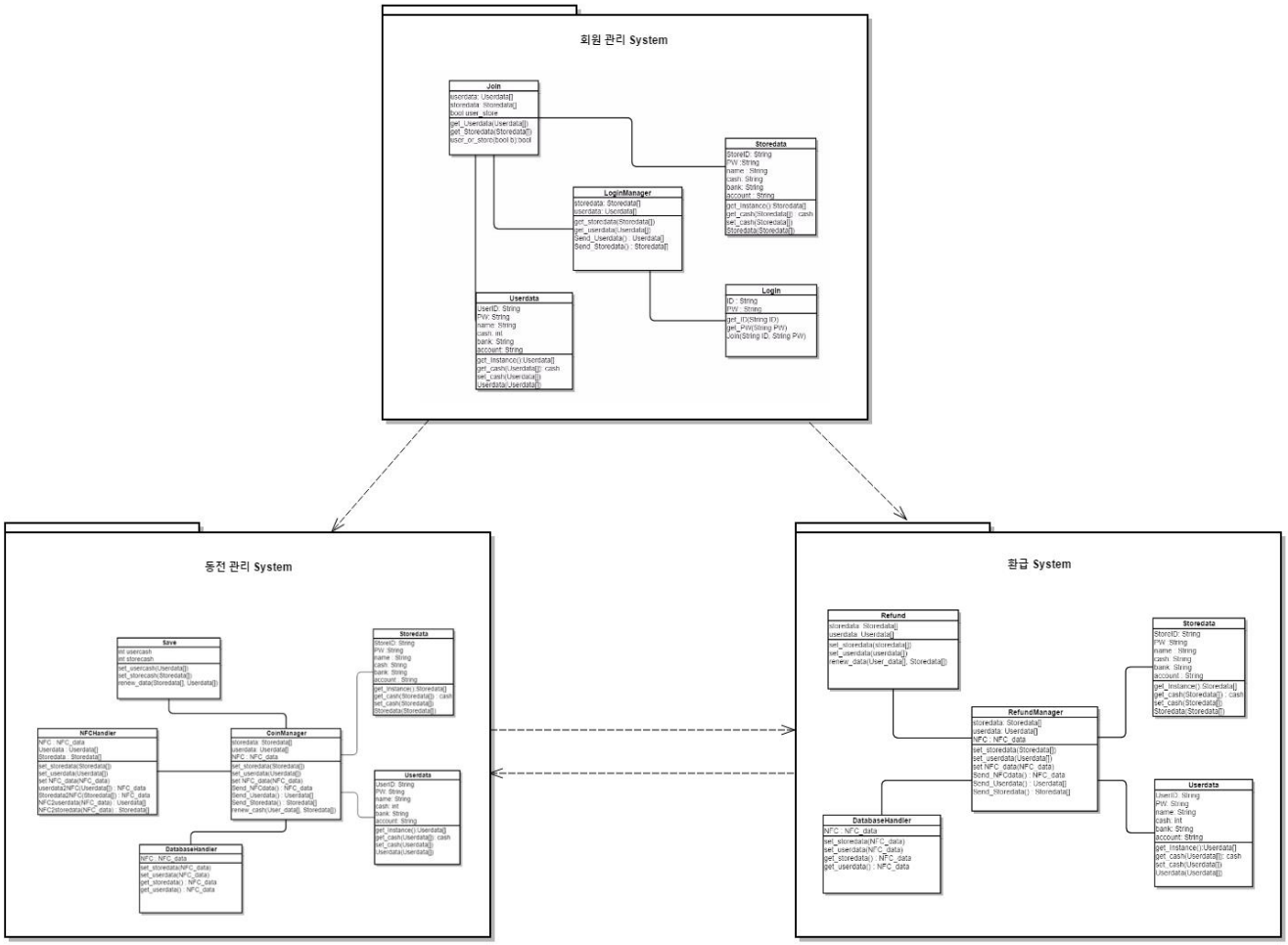

Diagram 9 Piggy-B 시스템 Package diagram

3.4 Deployment Diagram

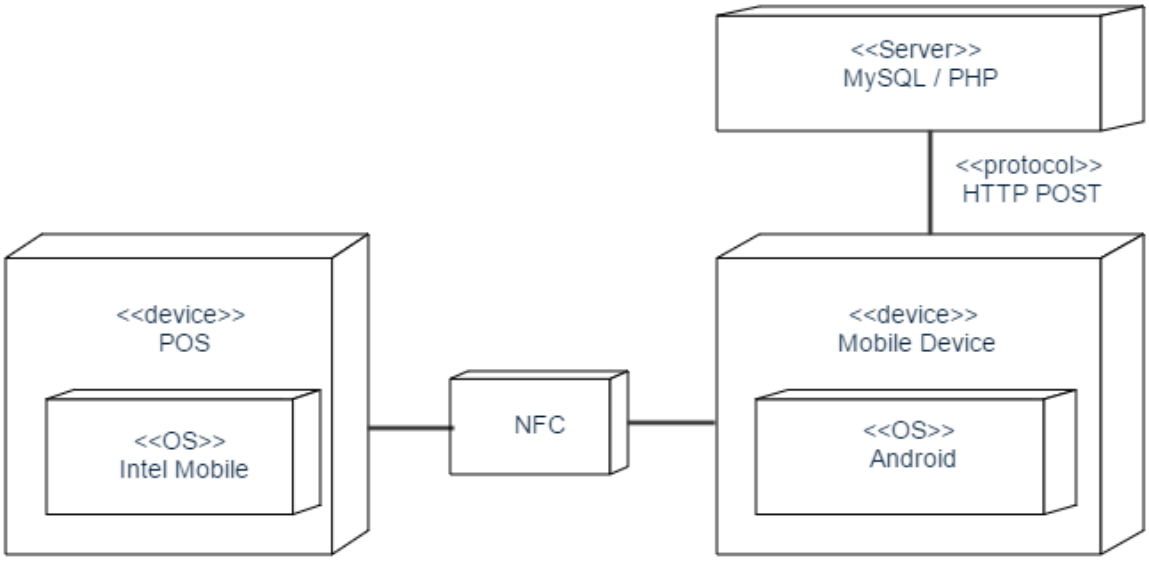

Diagram 10 Piggy-B 시스템 Deployment Diagram

4. Common Components

4.1 Objective

Common Components 에는 모든 서브시스템에서 공용으로 사용하는 컴포넌트에 대해 설명한다. Piggy-B 시스템은 사용자 관리 시스템, 동전 관리 시스템, 환급 시스템으로 나눌 수 있는데, 각 서브시스템에서 공용으로 사용하는 컴포넌트로는 Userdata, Storedata 가 있다.

4.2 Userdata

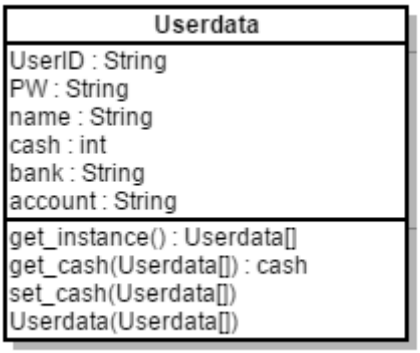

Diagram 11 Userdata Class Diagram

A Attributes

UserID: 고객의 ID를 저장하기 위해 사용한다.

PW: 비밀번호를 저장하기 위해 사용한다.

name: 이름을 저장하기 위해 사용한다.

bank: 이체할 은행을 저장하기 위해 사용한다.

account: 계좌번호를 저장하기 위해 사용한다.

B Operations

get_Instance() : Userdata []를 반환한다.

Userdata(Userdata []) : Userdata []를 만든다.

4.3 Storedata

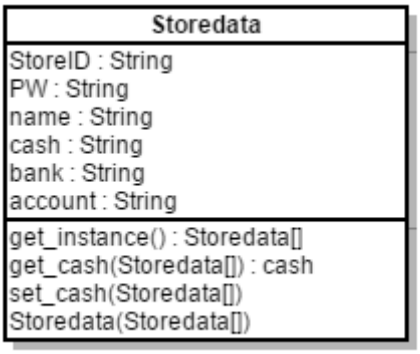

Diagram 12 Storedata Class Diagram

A Attributes

StoreID: 가맹점의 ID 를 저장하기 위해 사용한다.

PW: 비밀번호를 저장하기 위해 사용한다.

name: 이름을 저장하기 위해 사용한다.

bank: 이체할 은행을 저장하기 위해 사용한다.

account: 계좌번호를 저장하기 위해 사용한다.

B Operations

get_Instance() : Storedata []를 반환한다.

Storedata(Storedata []) : storedata []를 만든다.

5. User Activity

5.1 Class Diagram

A Login Activity

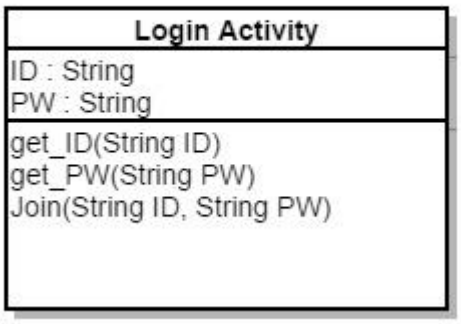

Diagram 13 Login Activity Class Diagram

A.1 Attributes

ID: 사용자가 자신의 아이디를 입력하는 UI 컴포넌트

PW: 사용자가 자신의 계정에 대한 비밀번호를 입력하는 UI 컴포넌트

A.2 Operations

get_ID(String ID): 액티비티가 실행되고 사용자의 ID를 서버에서 받아올 data로 변환한다.

get_PW(String PW): 액티비티가 실행되고 사용자의 PW를 서버에서 받아올 data로 변환한다.

Join(String ID, String PW): 사용자가 Join을 누르면 액티비티의 뷰 초기화를 진행

하고 로그인의 인증을 진행한다.

B Coin management Activity

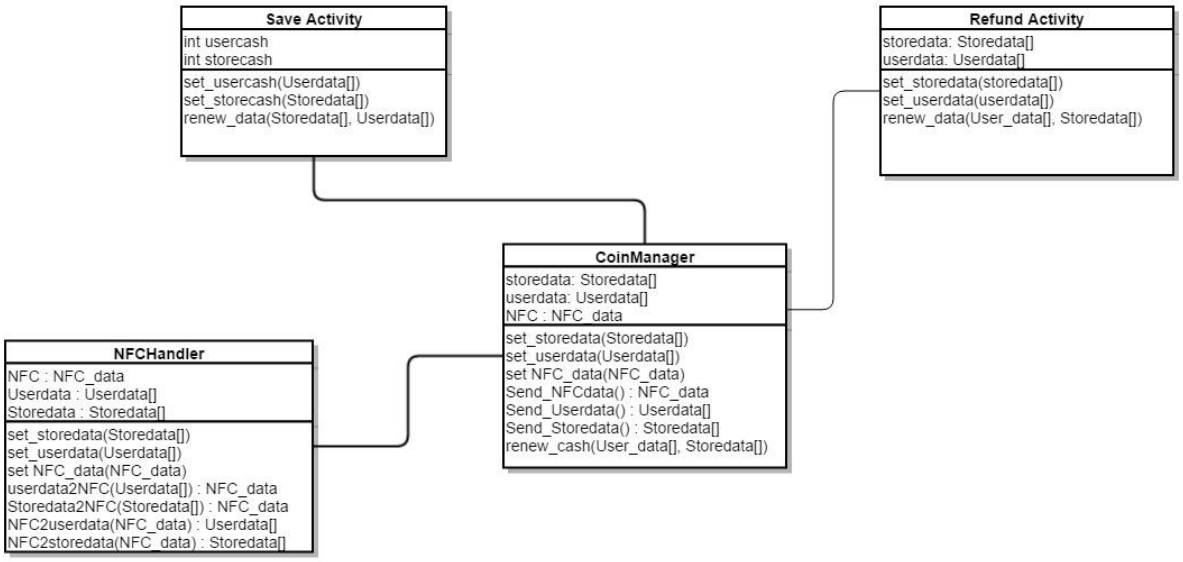

Diagram 14 Coin management Activity Class Diagram

B.1 Save Activity

B.1.1 Attributes

usercash: 가져온 고객의 적립금액을 저장하기 위해 사용한다.

storecash: 가져온 가맹점의 반환금액을 저장하기 위해 사용한다.

B.1.2 Operations

set_usercash(Userdata[]): Userdata[]를 가져와 usercash를 저장한다.

set_storecash(Storedata[]): Storedata[]를 가져와 storecash를 저장한다.

renew_data(Storedata[], Userdata[]): Userdata[], Storedata[]를 가져온 뒤 각

데이터를 갱신한다.

B.2 Refund Activity

B.2.1 Attributes

Userdata[]: 고객의 정보를 저장하기 위해 사용한다.

Storedata[]: 가맹점의 정보를 저장하기 위해 사용한다.

B.2.2 Operations

set_storedata(Storedata[]): storedata[]를 가져와 환급할 가맹점 금액에 저장하고 출력한다.

set_userdata(Userdata[]): Userdata[]를 가져와 환급할 고객의 금액에 저장하고 출력한다.

renew_data(Storedata[], Userdata[]): Userdata[], Storedata[]를 가져온 뒤 각 데이터를 갱신한다.

B.3 Coin Activity

B.3.1 Attributes

Userdata[]: 고객의 정보를 저장하고 보기 위해 사용한다.

Storedata[]: 가맹점의 정보를 저장하고 보기 위해 사용한다.

B.3.2 Operations

renew_data(Storedata[], Userdata[]): Userdata[], Storedata[]를 가져온 뒤 각 데이터를 갱신한다.

set_storedata(Storedata[]): storedata[]를 가져와 가맹점의 정보에 저장하고 출력한다.

set_userdata(Userdata[]): Userdata[]를 가져와 고객의 정보에 저장하고 출력한다.

5.2 Sequence Diagram

A Piggy-B 로그인 Sequence Diagram

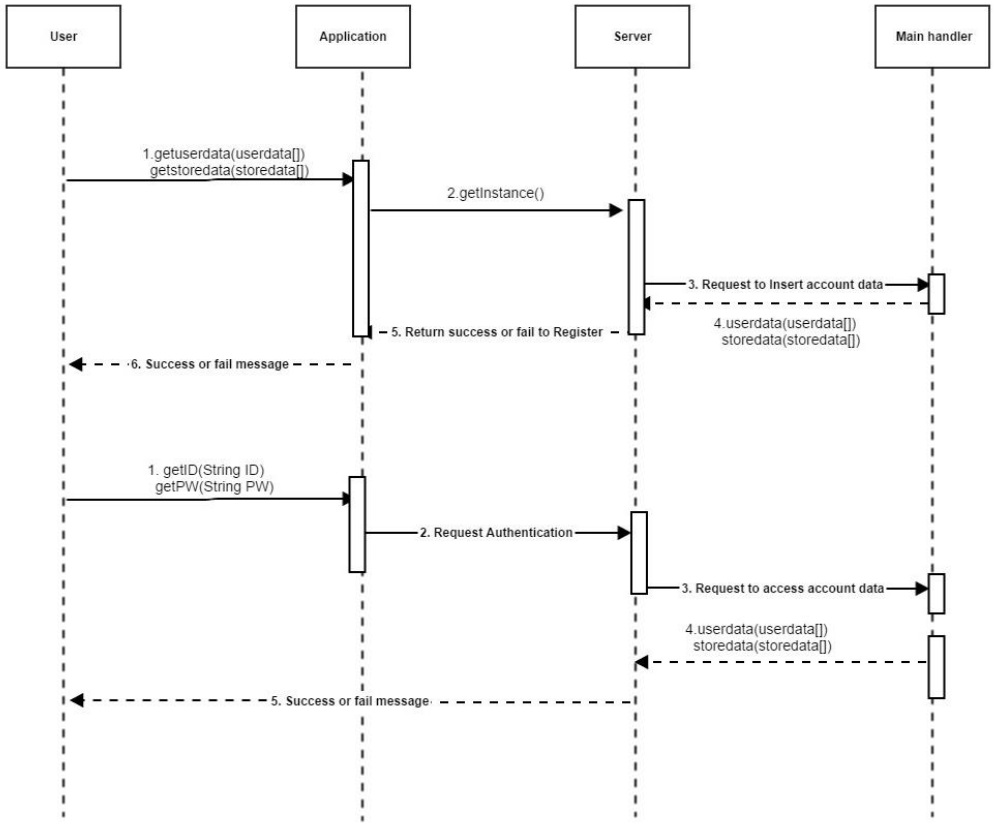

Diagram 15 Piggy-B 로그인 Sequence Diagram

Messages	Description
(1) getuserdata(Userdata[]) or getstoredata	고객의 정보를 입력 받는다.
(2) get_Instance()	해당 계정정보를 서버에 보내는 Instance 생성
(3) Insert to Userdata(Userdata[])	고객의 정보를 서버에 저장한다.
(4) send message	계정의 생성이 성공이면 2, 실패하면 3을 반환하고 해당 리턴 값에 맞는 메시지를 출력한다.
(5) getID(String ID) or getPW(String PW)	로그인 할 아이디와 패스워드를 입력 받는다.
(6) Request Authentication	로그인에 대한 인증을 요청한다.
(7) Request to access account data	해당 계정에 대한 접근을 요청한다.

Table 1 Piggy-B 로그인 Sequence Diagram의 Description

B 동전 적립 Sequence Diagram

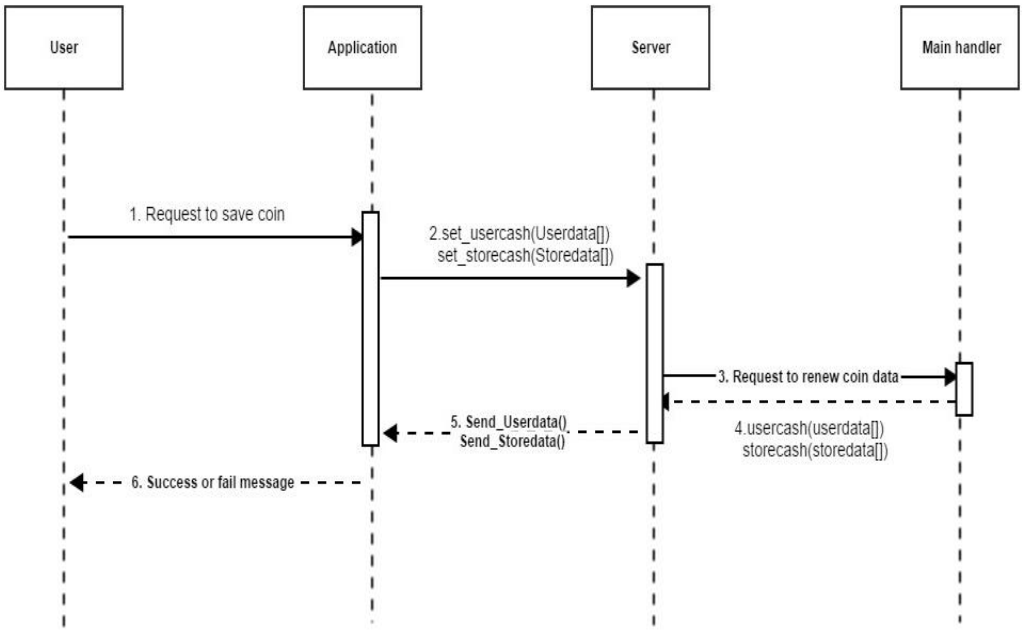

Diagram 16 동전 적립 Sequence Diagram

Messages	Description
(1) Request to save coin	사용자가 앱에 동전을 적립할 것을 요청
(2) set_usercash(Userdata[]) or set_storecash	해당 금액만큼의 동전을 set한다.
(3) Renew coin data	적립 금액을 서버에서 갱신한다.
(4) Usercash(userdata[]) or storecash	갱신된 금액을 set한다..
(5) send_Userdata() or send_storedata()	갱신된 금액을 앱에 보낸다.
(6) success or fail message	성공 시 갱신된 금액과 성공 메시지를 보여준다. 실패 시 실패 메시지를 보여준다.

Table 2 동전 적립 Sequence Diagram의 Description

C 환급 Sequence Diagram

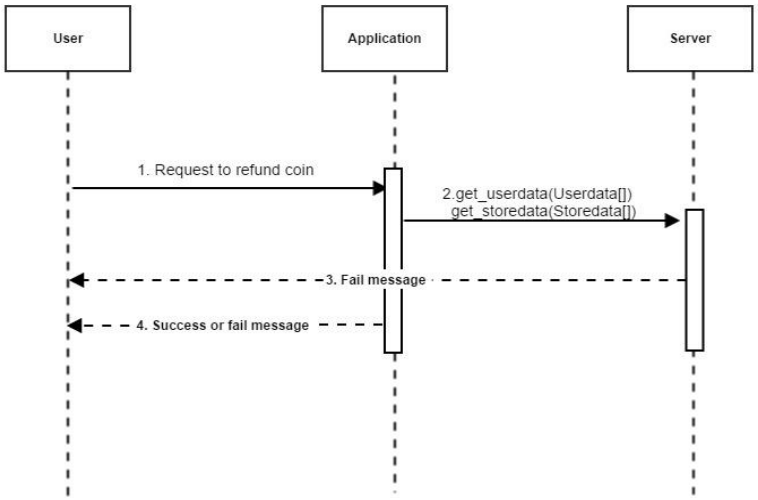

Diagram 17 환급 Sequence Diagram

Messages	Description
(1) Request to refund coin	사용자가 앱에 동전을 환급할 것을 요청
(2) get_usercash(Userdata[]) or get_storecash	해당 금액만큼의 동전을 가져온다.

(3) fail message	환급기준금액 미달 시 오류 메시지 전송
(4) success message	환급이 성공적으로 됐을 시 환급 금액과 성공 메시지 전송

Table 3 환급 Sequence Diagram의 Description

5.3 State Diagram

A Piggy-B 로그인 State Diagram

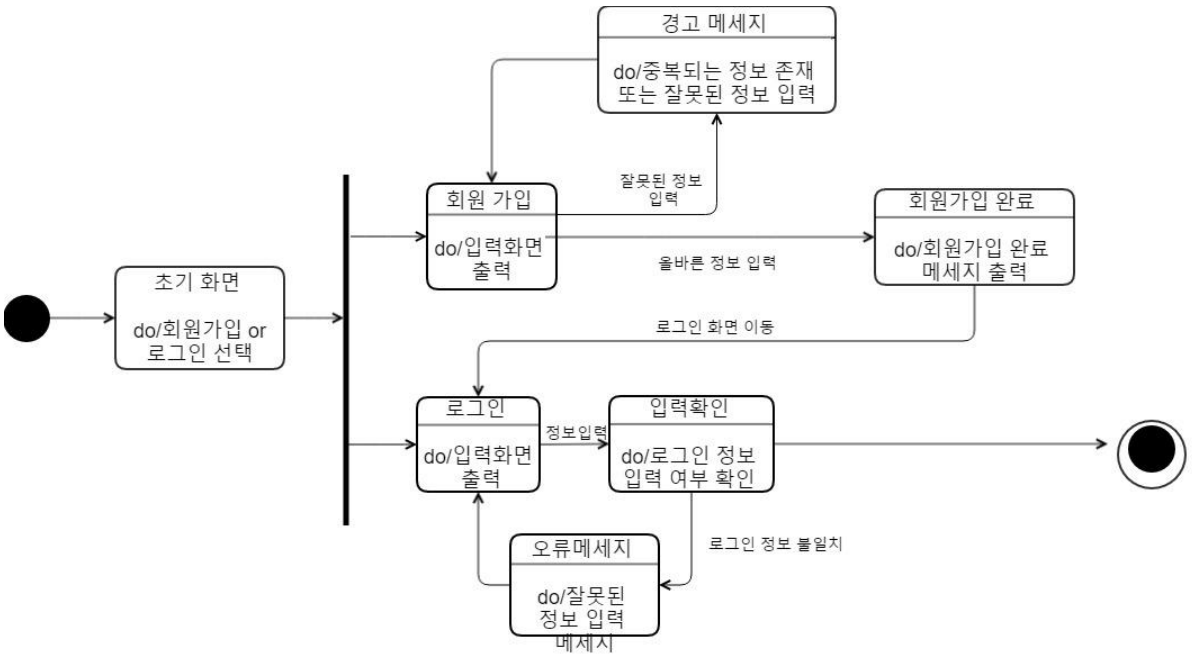

Diagram 18 Piggy-B 로그인 State Diagram

B 동전 적립 State Diagram

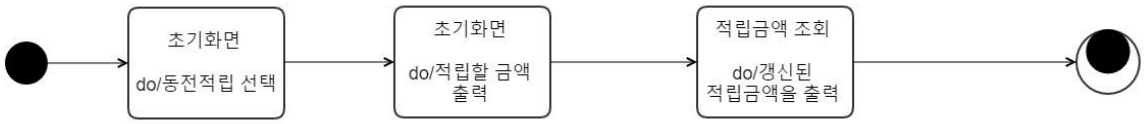

Diagram 19 동전 적립 State Diagram

C 환급 State Diagram

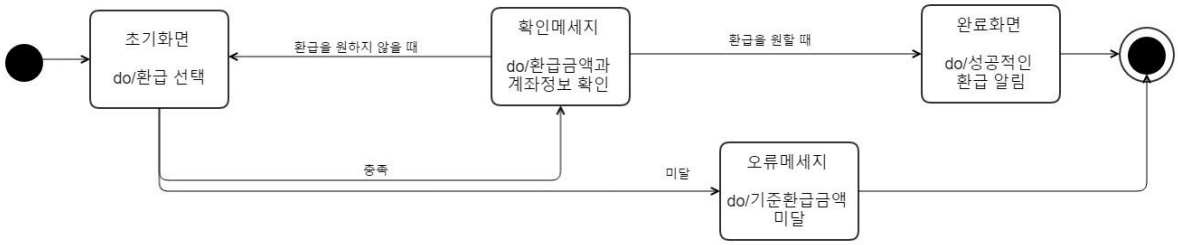

Diagram 20 환급 State Diagram

6. 사용자 관리 Subsystem

6.1 Class Diagram

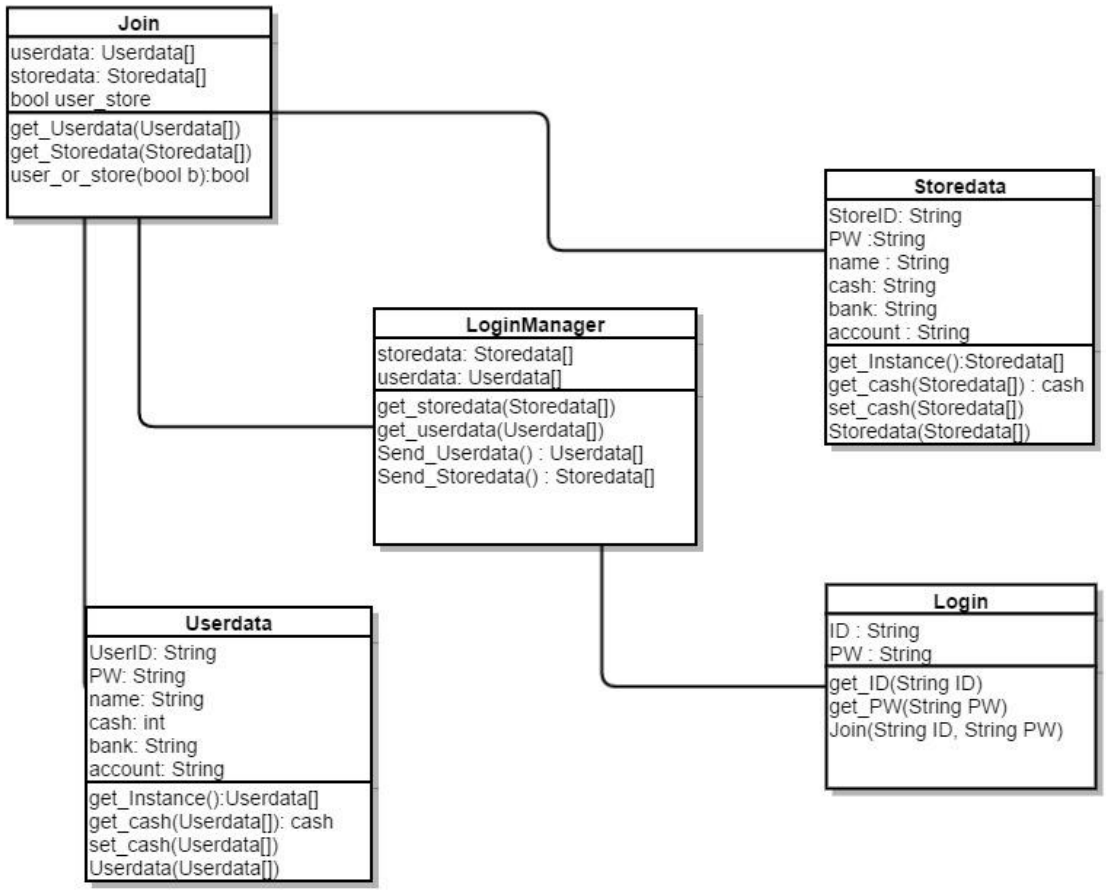

Diagram 21 사용자 관리 Subsystem Class Diagram

A Join

A.1 Attributes

userdata: 고객의 회원가입 및 정보를 위해 사용한다.

storedata: 가맹점의 회원가입 및 정보를 위해 사용한다.

user_store: 고객인지 가맹점인지 식별하기 위해 사용한다.

A.2 Operations

get_Userdata(Userdata[]): 고객의 정보를 입력 받는다.

get_Storedata(Storedata[]): 가맹점의 정보를 입력 받는다.

user_or_store(bool b): 고객인지 가맹점인지 0또는 1로 입력 받는다.

B Userdata

B.1 Attributes

UserID: 고객의 ID를 저장하기 위해 사용한다.

PW: 비밀번호를 저장하기 위해 사용한다.

name: 이름을 저장하기 위해 사용한다.

bank: 이체할 은행을 저장하기 위해 사용한다.

account: 계좌번호를 저장하기 위해 사용한다.

get_Instance(): Userdata[]를 반환한다.

Userdata(Userdata[]): Userdata[]를 만든다.

C Storedata

C.1 Attributes

StoreID: 가맹점의 ID를 저장하기 위해 사용한다.

PW: 비밀번호를 저장하기 위해 사용한다.

name: 이름을 저장하기 위해 사용한다.

bank: 이체할 은행을 저장하기 위해 사용한다.

account: 계좌번호를 저장하기 위해 사용한다.

C.2 Operations

get_Instance(): Storedata[]를 반환한다.

Storedata(Storedata[]): storedata[]를 만든다.

D LoginManager

D.1 Attributes

Userdata[]: 고객의 정보를 저장하기 위해 사용한다.

Storedata[]: 가맹점의 정보를 저장하기 위해 사용한다.

D.2 Operations

Send_Userdata(): Userdata를 전송한다.

Send_Storedata(): Storedata를 전송한다.

renew_data(Storedata[], Userdata[]): Userdata[], Storedata[]를 가져온 뒤 각 데이터를 갱신한다.

set_storedata(Storedata[]): storedata[]를 가져와 this.storedata[]에 저장한다.

set_userdata(Userdata[]): Userdata[]를 가져와 this.Userdata[]에 저장한다.

E Login

E.1 Attributes

ID: 사용자의 아이디를 입력 받기 위해 사용한다.

PW: 사용자의 비밀번호를 입력 받기 위해 사용한다.

E.2 Operations

getID(String ID): 사용자의 아이디를 입력 받는다.

getPW(String PW): 사용자의 비밀번호를 입력 받는다.

Join(String ID, String PW): 해당 아이디와 비밀번호를 접속 요청을 한다.

6.2 Sequence Diagram

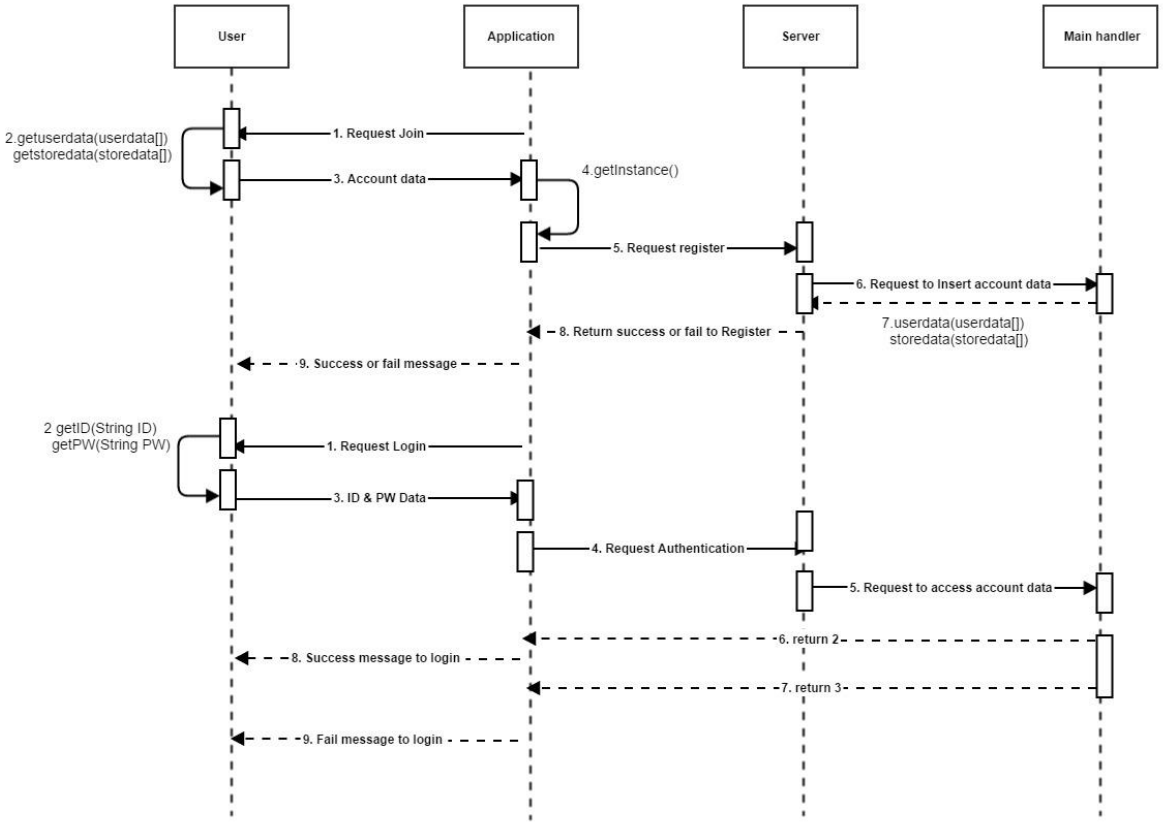

Diagram 22 사용자 관리 Subsystem Sequence Diagram

Messages	Description
(1) Request Join	회원가입이나 접속을 요청한다.
(2) getuserdata(Userdata[]) or getstoredata	고객의 정보를 입력 받는다.
(3) Account data	등록할 계정정보를 보낸다.
(4) get_Instance()	해당 계정정보를 서버에 보내는 Instance 생성
(5) Request Register	계정정보를 서버에 저장할 것을 요청한다.
(6) Insert to Userdata(Userdata[])	고객의 정보를 서버에 저장한다.
(7) send message	계정의 생성이 성공이면 2, 실패하면 3을 반환하고 해당 리턴 값에 맞는 메시지를 출력한다.
(8) Request Login	로그인을 요청한다.

(9) getID(String ID) or getPW(String PW)	로그인 할 아이디와 패스워드를 입력 받는다.
(10) Request Authentication	로그인에 대한 인증을 요청한다.
(11) Request to access account data	해당 계정에 대한 접근을 요청한다.

Table 4 사용자 관리 Subsystem Sequence Diagram의 Description

6.3 State Diagram

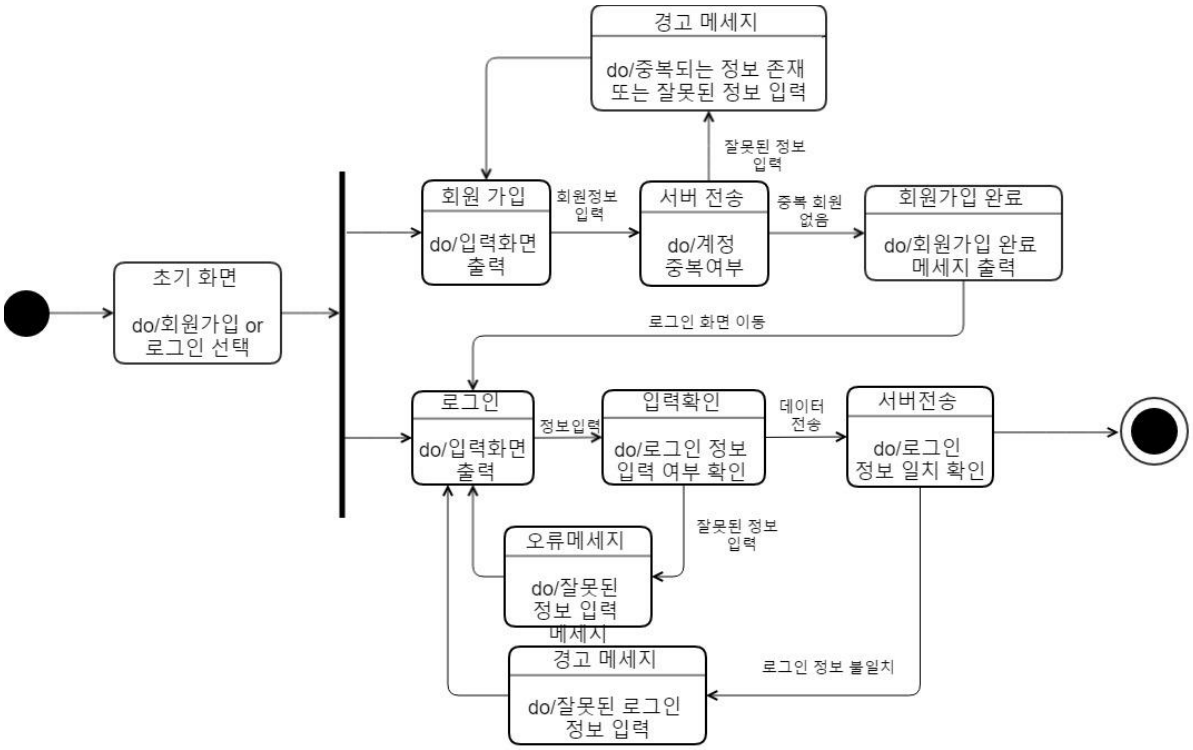

Diagram 23 사용자 관리 Subsystem State Diagram

7. 동전 관리 Subsystem

7.1 Class Diagram

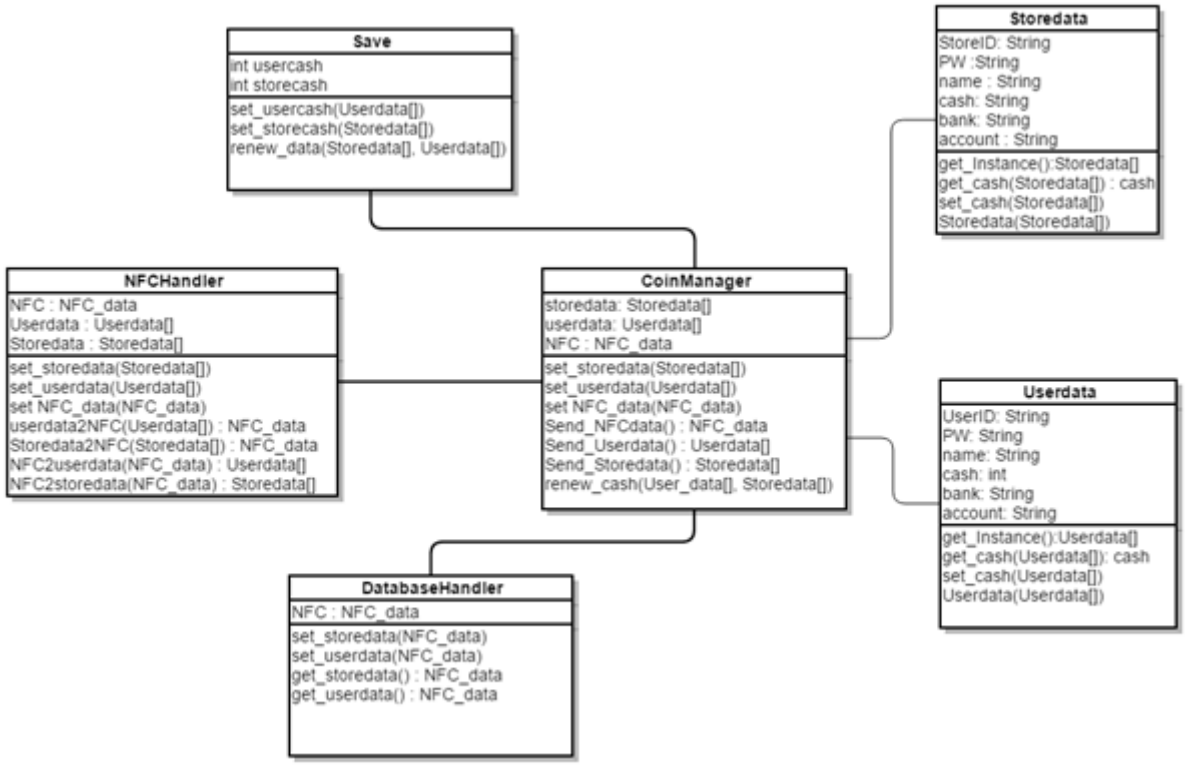

Diagram 24 동전 관리 Subsystem Class Diagram

A Save

A.1 Attributes

usercash: 가져온 고객의 적립금액을 저장하기 위해 사용한다.

storecash: 가져온 가맹점의 반환금액을 저장하기 위해 사용한다.

A.2 Operations

set_usercash(Userdata[]): Userdata[]를 가져와 usercash를 저장한다.

set_storecash(Storedata[]): Storedata[]를 가져와 storecash를 저장한다.

renew_data(Storedata[], Userdata[]): Userdata[], Storedata[]를 가져온 뒤 각 데이터를 갱신한다.

B NFCHandler

B.1 Attributes

NFC_data: NFC로 변환한 데이터를 저장하기 위해 사용한다.

Userdata[]: 고객의 정보를 저장하기 위해 사용한다.

Storedata[]: 가맹점의 정보를 저장하기 위해 사용한다.

B.2 Operations

set_storedata(Storedata[]): storedata[]를 가져와 this.storedata[]에 저장한다.

set_userdata(Userdata[]): Userdata[]를 가져와 this.Userdata[]에 저장한다.

set_NFC_data(NFC_data): NFC_data를 가져와 NFC_data에 저장한다.

userdata2NFC(Userdata[]): userdata[]를 가져와 NFC_data로 변환한다.

Storedata2NFC(Storedata[]): Storedata[]를 가져와 NFC_data로 변환한다.

NFC2userdata(NFC_data): NFC_data를 가져와 Userdata[]로 변환한다.

NFC2storedata(NFC_data): NFC_data를 가져와 Storedata[]로 변환한다.

C DatabaseHandler

C.1 Attributes

NFC_data: NFC로 변환한 데이터를 저장하기 위해 사용한다.

C.2 Operations

set_storedata(NFC_data): 서버에서 가맹점에 관한 NFC_data를 받아 저장한다.

set_userdata(NFC_data): 서버에서 고객에 관한 NFC_data를 받아 저장한다.

get_storedata() : 가맹점에 관한 NFC_data를 반환한다.

get_userdata() : 고객에 관한 NFC_data를 반환한다.

D CoinManager

D.1 Attributes

NFC_data: NFC로 변환한 데이터를 저장하기 위해 사용한다.

Userdata[]: 고객의 정보를 저장하기 위해 사용한다.

Storedata[]: 가맹점의 정보를 저장하기 위해 사용한다.

D.2 Operations

Send_NFCdata(): NFC_data를 전송한다.

Send_Userdata(): Userdata를 전송한다.

Send_Storedata(): Storedata를 전송한다.

renew_data(Storedata[], Userdata[]): Userdata[], Storedata[]를 가져온 뒤 각 데이터를 갱신한다.

set_storedata(Storedata[]): storedata[]를 가져와 this.storedata[]에 저장한다.

set_userdata(Userdata[]): Userdata[]를 가져와 this.Userdata[]에 저장한다.

set_NFC_data(NFC_data): NFC_data를 가져와 NFC_data에 저장한다.

E Storedata

E.1 Attributes

StoreID: 가맹점의 ID를 저장하기 위해 사용한다.

PW: 비밀번호를 저장하기 위해 사용한다.

name: 이름을 저장하기 위해 사용한다.

cash: 금액을 저장하기 위해 사용한다.

bank: 이체할 은행을 저장하기 위해 사용한다.

account: 계좌번호를 저장하기 위해 사용한다.

E.2 Operations

get_Instance(): Storedata[]를 반환한다.

get_cash(Storedata[]): cash를 반환한다.

set_cash(Storedata[]): cash를 저장한다.

Storedata(Storedata[]): storedata[]를 만든다.

F Userdata

F.1 Attributes

UserID: 고객의 ID를 저장하기 위해 사용한다.

PW: 비밀번호를 저장하기 위해 사용한다.

name: 이름을 저장하기 위해 사용한다.

cash: 금액을 저장하기 위해 사용한다.

bank: 이체할 은행을 저장하기 위해 사용한다.

account: 계좌번호를 저장하기 위해 사용한다.

F.2 Operations

get_Instance(): Userdata[]를 반환한다.

get_cash(Userdata[]): cash를 반환한다.

set_cash(Userdata[]): cash를 저장한다.

Userdata(Userdata[]): Userdata[]를 만든다.

7.2 Sequence Diagram

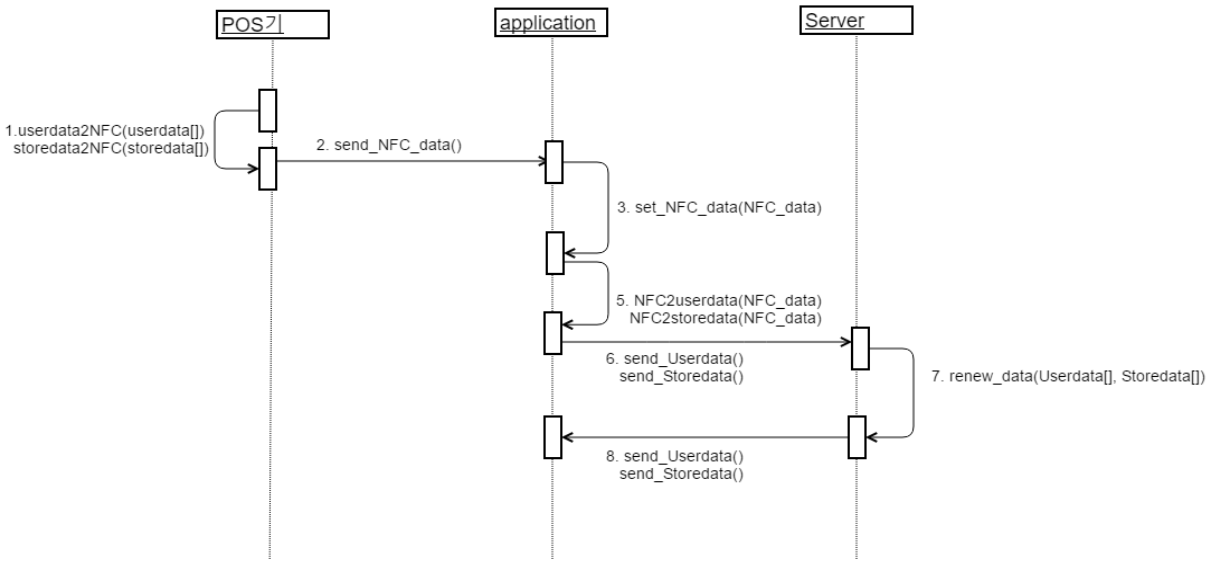

Diagram 25 동전 관리 Subsystem Sequence Diagram

Messages	Description
(1) userdata2NFC(userdata[])	userdata[]를 NFC_data로 바꿔준다.

(2) storedata2NFC(userdata[])	storedata[]를 NFC_data로 바꿔준다.
(3) send_NFC_data()	NFC_data를 전송한다.
(4) set_NFC_data(NFC_data)	NFC_data를 받아 저장한다.
(5) NFC2userdata(NFC_data)	NFC_data를 Userdata[]로 바꿔준다.
(6) NFC2storedata(NFC_data)	NFC_data를 Storedata[]로 바꿔준다.
(7) send_Userdata()	Userdata를 전송한다.
(8) send_storedata()	Storedata를 전송한다.
(9) renew_data(Userdata[], Storedata[])	userdata[]와 storedata[]의 금액을 갱신한다.

Table 5 동전 관리 Subsystem Sequence Diagram의 Description

7.3 State Diagram

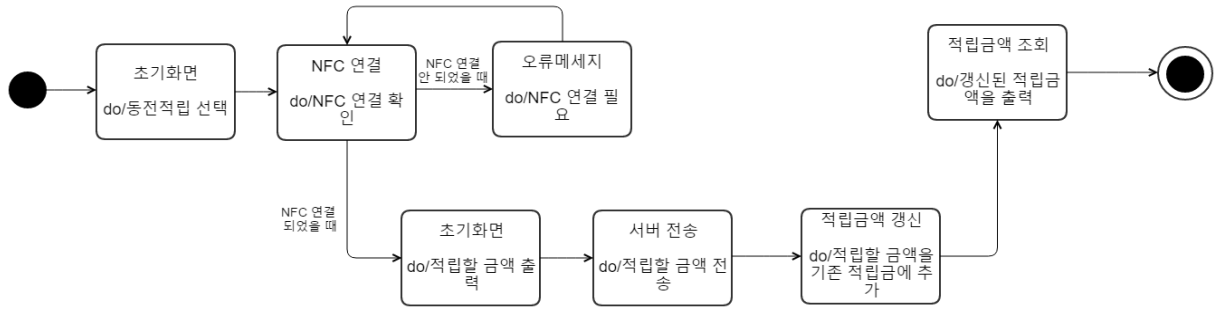

Diagram 26 동전 관리 Subsystem State Diagram

8. 환급 Subsystem

8.1 Class Diagram

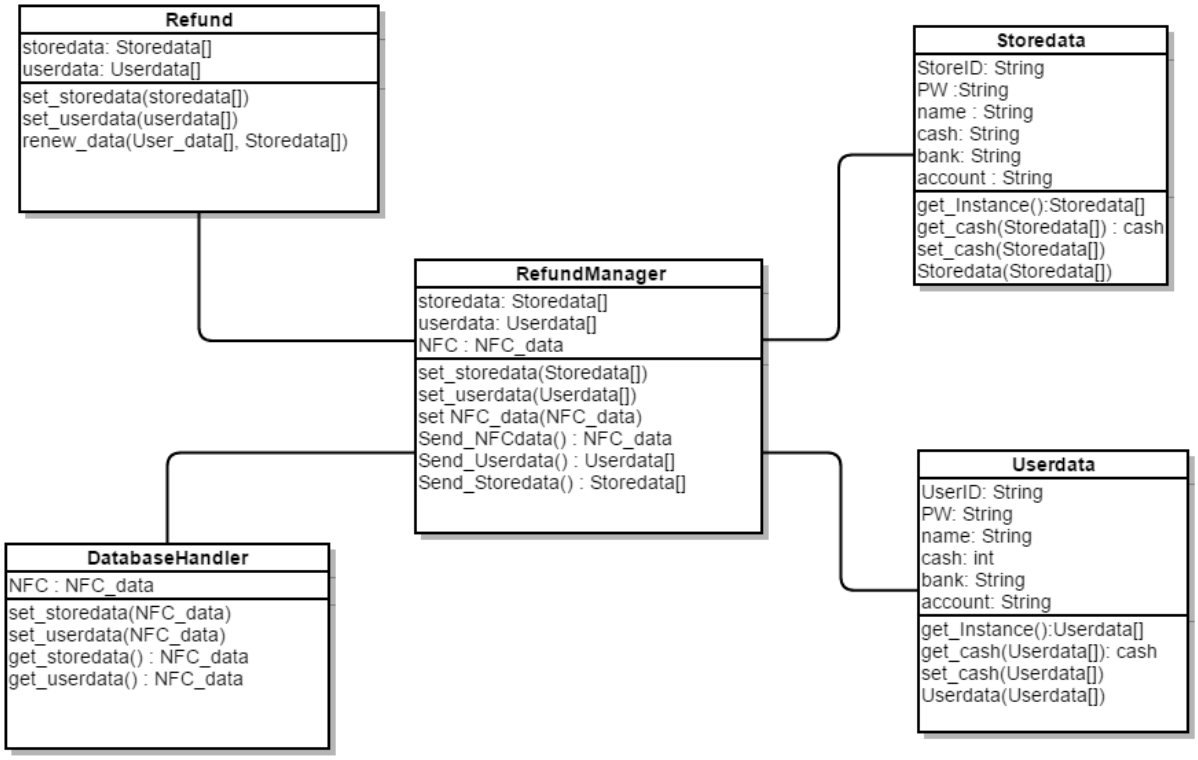

Diagram 27 환급 Subsystem Class Diagram

A RefundManager

A.1 Attributes

NFC_data: NFC로 변환한 데이터를 저장하기 위해 사용한다.

Userdata[]: 고객의 정보를 저장하기 위해 사용한다.

Storedata[]: 가맹점의 정보를 저장하기 위해 사용한다.

A.2 Operations

Send_NFCdata(): NFC_data를 전송한다.

Send_Userdata(): Userdata를 전송한다.

Send_Storedata(): Storedata를 전송한다.

set_storedata(Storedata[]): storedata[]를 가져와 this.storedata[]에 저장한다.

set_userdata(Userdata[]): Userdata[]를 가져와 this.Userdata[]에 저장한다.

set_NFC_data(NFC_data): NFC_data를 가져와 NFC_data에 저장한다.

B Refund

B.1 Attributes

Userdata[]: 고객의 정보를 저장하기 위해 사용한다.

Storedata[]: 가맹점의 정보를 저장하기 위해 사용한다

B.2 Operations

set_storedata(Storedata[]): storedata[]를 가져와 this.storedata[]에 저장한다.

set_userdata(Userdata[]): Userdata[]를 가져와 this.Userdata[]에 저장한다.

renew_data(Storedata[], Userdata[]): Userdata[], Storedata[]를 가져온 뒤 각 데이터를 갱신한다.

C DatabaseHandler

C.1 Attributes

NFC_data: NFC로 변환한 데이터를 저장하기 위해 사용한다.

C.2 Operations

set_storedata(NFC_data): 서버에서 가맹점에 관한 NFC_data를 받아 저장한다.

set_userdata(NFC_data): 서버에서 고객에 관한 NFC_data를 받아 저장한다.

get_storedata() : 가맹점에 관한 NFC_data를 반환한다.

get_userdata() : 고객에 관한 NFC_data를 반환한다.

D Userdata

D.1 Attributes

UserID: 고객의 ID를 저장하기 위해 사용한다.

PW: 비밀번호를 저장하기 위해 사용한다.

name: 이름을 저장하기 위해 사용한다.

cash: 금액을 저장하기 위해 사용한다.

bank: 이체할 은행을 저장하기 위해 사용한다.

account: 계좌번호를 저장하기 위해 사용한다.

D.2 Operations

get_Instance(): Userdata[]를 반환한다.

get_cash(Userdata[]): cash를 반환한다.

set_cash(Userdata[]): cash를 저장한다.

Userdata(Userdata[]): Userdata[]를 만든다.

E Storedata

E.1 Attributes

StoreID: 가맹점의 ID를 저장하기 위해 사용한다.

PW: 비밀번호를 저장하기 위해 사용한다.

name: 이름을 저장하기 위해 사용한다.

cash: 금액을 저장하기 위해 사용한다.

bank: 이체할 은행을 저장하기 위해 사용한다.

account: 계좌번호를 저장하기 위해 사용한다.

E.2 Operations

get_Instance(): Storedata[]를 반환한다.

get_cash(Storedata[]): cash를 반환한다.

set_cash(Storedata[]): cash를 저장한다.

Storedata(Storedata[]): storedata[]를 만든다.

8.2 Sequence Diagram

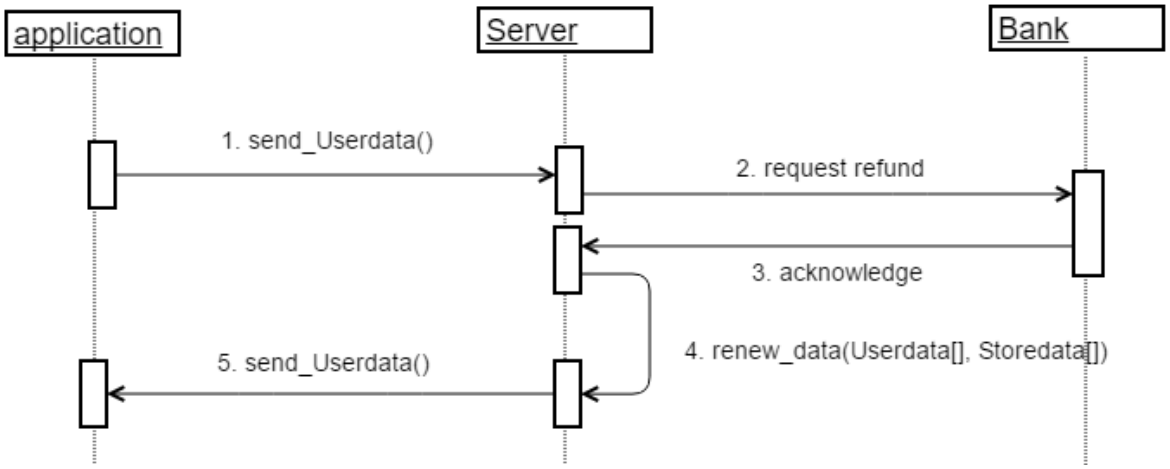

Diagram 28 환급 Subsystsem Sequence Diagram

Messages	Description
(1) send_Userdata()	Userdata를 전송한다.
(2) request refund	은행에 환급을 요청한다.
(3) acknowledge	은행에서 요청을 받았음을 알린다.
(4) renew_data(Userdata[], Storedata[])	userdata[]와 storedata[]의 금액을 갱신한다.

Table 6 환급 Subsystem Sequence Diagram의 Description

8.3 State Diagram

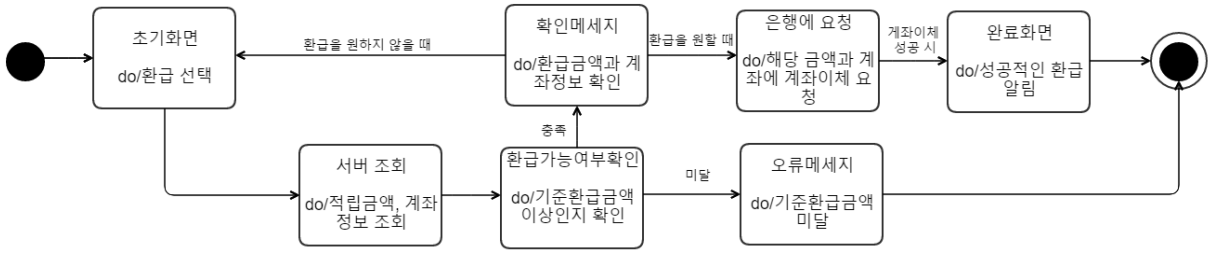

Diagram 29 환급 Subsystem State Diagram

9. Communication Protocol

9.1 Objectives

Protocol Design에서는 서브 시스템들이 상호 통신하기 위하여 약속해놓은 프로토콜(protocol) 규약에 대해 서술한다. 프로토콜의 기본 형식은 JSON을 기본으로 하며, 통신하는 메시지(message)의 형식과 용도, 의미를 설명한다.

9.2 JSON

Figure 6 JSON

JSON은 JavaScript Object Notation의 약어로, JavaScript에 기반을 두고 있다. 객체의 형태를 표현하기 위하여 적용된 데이터 교환 방식이다. 간단한 데이터를 XML보다 더 쉽게 표현할 수 있으며, 기능도 적어서 파싱(parsing)이 빠르다는 장점을 가지고 있다. 클라이언트의 입장에서 매우 유용하며, 모바일을 기반으로 하는 데이터 전송에 적절한 형식을 가지고 있다. AJAX를 사용하며 데이터를 주고 받을 때의 포맷이기도 하다.

9.3 공통 프로토콜

A URL

모든 요청은 다음 URL로 전송되어 서버 커뮤니케이션 모듈이 처리하도록 한다.

요청 URL: piggg.comxa.com

B 공통 요청 변수

모든 요청은 다음 변수를 요청에 포함시켜야 한다.

변수명	타입	설명	비고
type	string	프로토콜 타입	각 요청마다 고유값을 가짐
deviceID	string	스마트폰 또는 POS기 주소	각 기기마다 고유값을 가짐

Table 7 공통 프로토콜 요청 변수

C 공통 응답 변수

서버는 다음 변수를 응답에 포함시켜야 한다.

변수명	타입	설명	비고
return	integer	응답 결과	처리 성공 시 2 처리 실패 시 3 오류 발생 시 4

Table 8 공통 프로토콜 응답 변수

9.4 사용자 관리 시스템 프로토콜

A 계정이 없을 때

A.1 요청 변수

변수명	타입	설명	비고
Type	string	프로토콜 타입	고정값 ”JoinStatus”
deviceID	string	스마트폰 또는 POS기 주소	
Identifier	Boolean	고객인지 상점인지 구별	0이면 고객 1이면 상점

Table 9 계정이 없을 때 요청 변수

A.2 서버 응답

변수명	타입	설명	비고
Return	Integer	응답 결과	

Table 10 계정이 없을 때 서버 응답

B 계정이 있을 때

B.1 요청 변수

변수명	타입	설명	비고
Type	String	프로토콜 타입	고정값 “Loginstatus”
DeviceID	String	스마트폰 또는 POS기 주소	
Identifier	Boolean	고객인지 상점인지 구별	0이면 고객 1이면 상점

Table 11 계정이 있을 때 요청 변수

B.2 서버 응답

변수명	타입	설명	비고

Return	Integer	응답결과	

Table 12 계정이 있을 때 서버 응답

9.5 동전 관리 시스템 프로토콜

A 요청 변수

변수명	타입	설명	비고
Type	String	프로토콜 타입	고정값 “coinmanage”
NFC_data	String	적립할 금액의 대한 NFC를 통한 정보	
DeviceID	String	스마트폰 또는 POS기 주소	

Table 13 동전 관리 시스템 프로토콜 요청 변수

B 서버 응답

변수명	타입	설명	비고
Return	integer	응답결과	
Coin_data	Integer	적립금액	기존의 적립금액에 적립할 금액을 더하여 갱신한 금액

User_data	String	고객 또는 가맹점의 계정 정보	고객의 정보 구성
			{
			“name”: string
			}

Table 14 동전 관리 시스템 프로토콜 서버 응답

9.6 환급 시스템 프로토콜

A 요청 변수

변수명	타입	설명	비고
Type	String	프로토콜 타입	고정값 “Loginstatus”
DeviceID	String	스마트폰 또는 POS기 주소	
Identifier	Boolean	환급이 가능한지 여부	0이면 환급 가능 1이면 환급 불가능

Table 15 환급 시스템 프로토콜 요청 변수

B 서버 응답

변수명	타입	설명	비고
Return	integer	응답결과	
refund_data	Integer	환급금액	환급할 금액에 대한 정보

User_data	String	고객 또는 가맹점의 계정 정보	고객의 정보 구성
			{
			“name”: string
			“bank”: string
			“account”: string
			}

Table 16 환급 시스템 프로토콜 서버 응답

10. Database Design

10.1 Objectives

Database Design에서는 요구사항 명세서에서 기술하였던 요구사항을 기반으로 데이터베이스를 설계하고 설명한다. 요구사항에 기반한 데이터베이스의 ER diagram을 제시하고, 이에 대한 relational schema를 서술한다. Normalization 과정을 통하여 데이터베이스 간에 존재할 수 있는 중복을 방지하고, SQL DDL을 작성한다.

10.2 ER Diagram

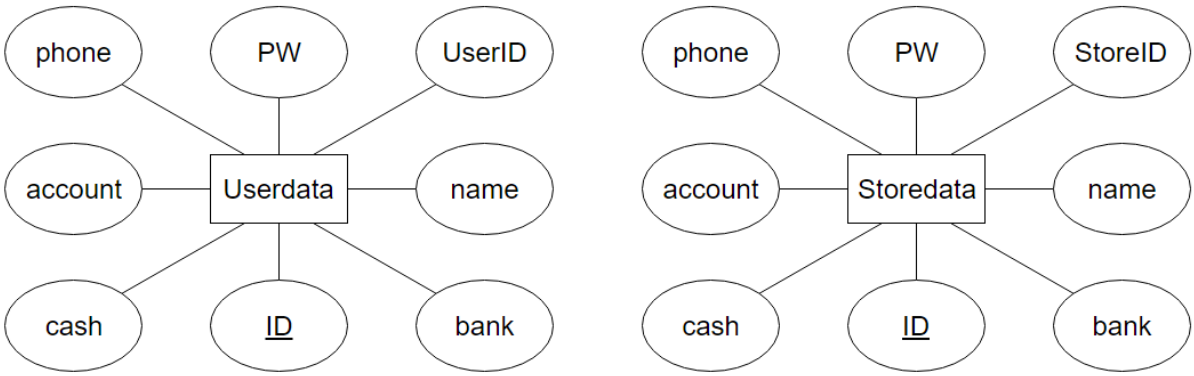

Diagram 30 Piggy-B ER Diagram

A Entity

A.1 User

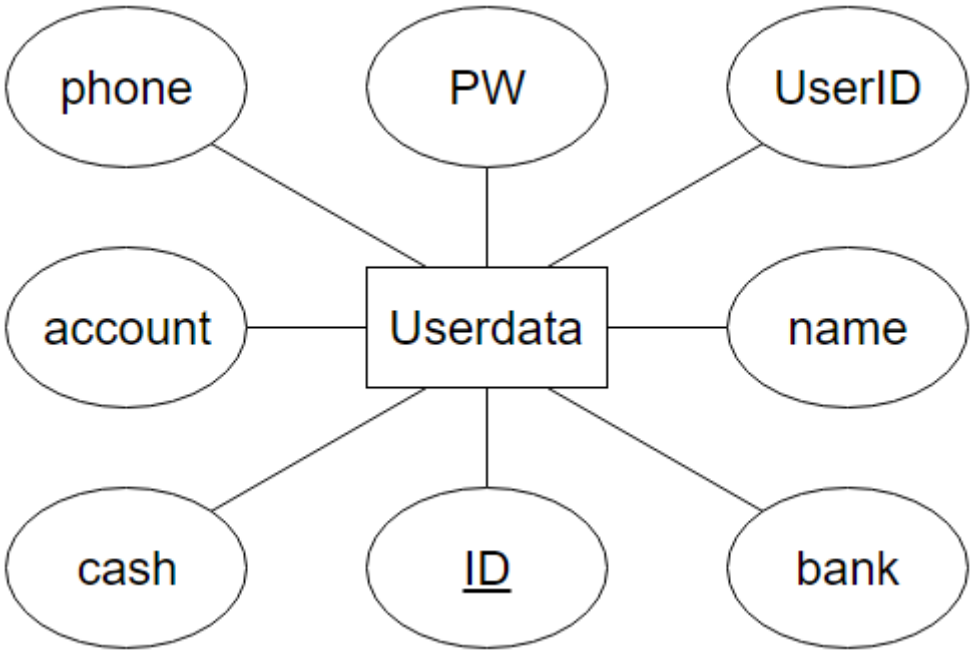

Diagram 31 Userdata ER Diagram

Userdata는 고객의 정보를 나타낸다. UserID, PW, phone, bank, account, name, cash는 고객의 ID, 비밀번호, 전화번호, 계좌가 있는 은행의 이름, 계좌번호, 이름, 잔액을 의미한다. ID는 index의 역할을 하며 Key이다.

A.2 Store

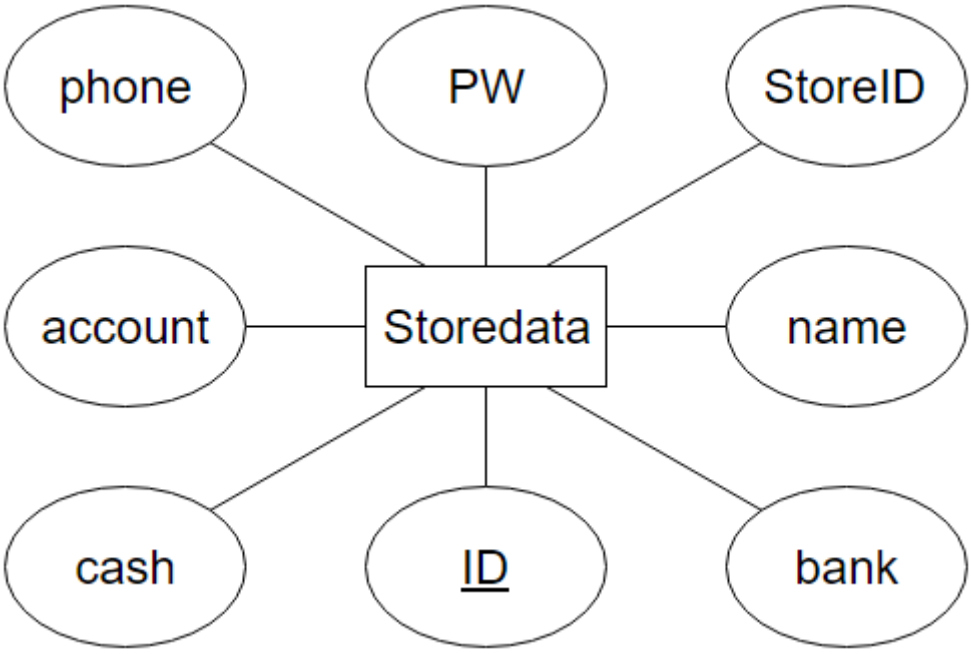

Diagram 32 Storedata ER Diagram

Storedata는 고객의 정보를 나타낸다. StoreID, PW, phone, bank, account, name, cash는 가맹점의 ID, 비밀번호, 전화번호, 계좌가 있는 은행의 이름, 계좌번호, 이름, 잔액을 의미한다. ID는 index의 역할을 하며 Key이다.

B Relationship

Piggy-B 시스템에는 Relationship이 없으므로 생략한다.

10.3 Relational Schema

A Userdata

Userdata							
ID	UserID	PW	phone	bank	account	name	cash

Diagram 33 Userdata Relational Schema

Primary key: ID

Functional Dependencies: ID-〉{UserID, PW, phone, bank, account, name, cash}

고객의 ID, 비밀번호, 전화번호, 계좌가 있는 은행의 이름, 계좌번호, 이름, 잔액을 담고 있는 테이블이다. ID가 key이므로 {UserID, PW, phone, bank, account, name, cash}를 결정한다.

B Storedata

Storedata							
ID	StoreID	PW	phone	bank	account	name	cash

Diagram 34 Storedata Relational Schema

Primary key: ID

Functional Dependencies: ID-〉{StoreID, PW, phone, bank, account, name, cash}

가맹점의 ID, 비밀번호, 전화번호, 계좌가 있는 은행의 이름, 계좌번호, 이름, 잔액을 담고 있는 테이블이다. ID가 key이므로 {StoreID, PW, phone, bank, account, name, cash}를 결정한다.

10.4 Normalization

정규화는 나쁜 테이블을 더 좋은 작은 테이블로 분해하는 것을 말한다. 나쁜 테이블은 중복을 갖고 있으며, 이 중복은 데이터베이스에 이상을 가져온다. 그 이상은 다음과 같은 것들이 있다.

Redundancy: 테이블에서 필요 이상의 정보가 여러 번 반복되는 것을 의미한다.

Insert Anomaly: 새로운 튜플을 삽입할 수 없다.

Delete Anomaly: 어떤 정보를 삭제할 때, 원치 않는 정보를 삭제하게 된다.

Update Anomaly: 정보를 업데이트 하고자 할 때, 여러 번 업데이트 해야 한다.

이를 해결하는 정규화에는 여러가지가 있는데, 제 1 정규화부터 최근에는 제 6 정규화까지 나왔다. 하지만 실제로 대다수 상용 데이터베이스 설계에서는 BCNF까지의 정규형만을 고려한다. 따라서 이 프로젝트에서는 BCNF를 목표로 하여 데이터베이스를 설계한다.

BCNF란 주어진 테이블 R에 대해서, 모든 functional dependency X->A에 대하여 X가 super key인 것을 의미한다. 따라서 각각의 테이블에서 모든 functional dependency X->A에 대하여 X가 super key인지 보고자 한다.

A Userdata

Functional Dependencies: ID-> {UserID, PW, phone, bank, account, name, cash}

설명: ID가 Super key이므로 BCNF를 만족한다.

B Storedata

Functional Dependencies: ID-> {StoreID, PW, phone, bank, account, name, cash}

설명: ID가 Super key이므로 BCNF를 만족한다.

10.5 SQL DDL

A Userdata

Userdata

CREATE TABLE Userdata{
ID INT NOT NULL.
UserID VARCHAR(32) NOT NULL,
PW VARCHAR(32) NOT NULL,
Phone VARCHAR(16) NOT NULL,
bank VARCHAR(16) NOT NULL,
account VARCHAR(16) NOT NULL,
name VARCHAR(32) NOT NULL,
cash INT,
PRIMARY KEY(UserID)
};

ID는 PRIMARY KEY이므로 NOT NULL로 선언하였고, cash를 제외한 나머지 정보도 필수 정보이므로 NOT NULL로 선언하였다. cash는 필수 정보이긴 하나, 고객 등록 후에 입력하는 정보이므로 NOT NULL로 선언하지 않았다.

Diagram 35 Userdata SQL DDL

B Storedata

Storedata
CREATE TABLE Storedata{ ID INT NOT NULL.

```
        StoreID VARCHAR(32) NOT NULL,
        PW      VARCHAR(32) NOT NULL,
        phone   VARCHAR(16) NOT NULL,
        bank    VARCHAR(16) NOT NULL,
        account VARCHAR(16) NOT NULL,
        name    VARCHAR(32) NOT NULL,
        cash    INT,
        PRIMARY KEY(StoreID)
};
```

ID는 PRIMARY KEY이므로 NOT NULL로 선언하였고, cash를 제외한 나머지 정보도 필수 정보이므로 NOT NULL로 선언하였다. cash는 필수 정보이긴 하나, 가맹점 등록 후에 입력하는 정보이므로 NOT NULL로 선언하지 않았다.

Diagram 36 Storedata SQL DDL

11. Testing Plan

11.1 Testing Policy

Piggy-B 시스템의 개발에서는 크게 세 단계로 테스트를 한다. Development Testing, Release testing, User testing으로 나뉘어지며, Development testing은 다시 component testing, integrating testing, system testing, acceptance testing의 네 단계로 나뉘어진다.

A Development Testing

Development testing은 개발을 하며 시스템의 문제를 찾아내는 작업이다.

B Component Testing

Component testing은 시스템을 component단위로 테스트하는 것으로, 각 구성요소들을 개발한 후에 제대로 작동하는지 확인하는 것이다. Integrating testing 은 시스템의 구성요소들을 개발하고 테스트 한 이후에 구성요소들을 하나씩 점진적으로 합치면서 하는 테스트이다. 각 구성 요소간의 interface와, 문제가 어디서 일어나는지 확인할 수 있다. System testing은 모든 sub-system을 하나로 합친 후에 시스템이 잘 동작하는지 테스트하는 것이다. 마지막으로 Acceptance testing은 사용자의 정보를 이용하여 시스템에 대한 사용자의 요구사항을 테스트하는 것이다.

C Release Testing

Release testing은 사용자에게 출시하는 최종 시스템을 출시 전에 테스트 하는 것이다.

D User Testing

User testing은 사용자가 사용자의 환경 및 관점에서 시스템을 테스트 하는 것을 말한다.

11.2 Test Case

A 사용자 관리

A.1 고객용 시스템

1) 사용자: 모바일 기기에 모바일 앱을 다운받는다.

2) 사용자: 모바일 앱을 실행한다.

3) 모바일 앱: “ID”, “Password”, “회원가입”을 입력하는 창을 출력한다.

4) 사용자: 계정 유무확인 또는 회원가입을 한다.

4.1) 계정이 있을 시 - ID와 Password를 입력하고 Join 버튼을 누른다.

4.2) 계정이 없을 시 - 고객의 계정 정보를 입력하는 회원가입 화면을 출력한다.

4.2.1) 모바일 앱: 고객의 계정 정보를 데이터베이스에 보낸다.

4.2.2) 시스템 동작: 해당 계정이 데이터베이스에 등록이 되는지 확인한다.

4.2.2.1) 등록 가능 - 해당 계정을 데이터베이스에 저장한다.

4.2.2.2) 등록 실패 - 모바일 앱: “회원 가입 실패” 메시지를 출력한다.

5) 모바일 앱: ID와 Password가 일치하는지 확인한다.

5.1) 일치 - 모바일 앱: 해당 계정 정보를 데이터베이스에서 가져와 출력한다.

5.2 불일치 - 모바일 앱: “로그인 실패” 메시지를 출력한다.

6) 시스템 동작: 계정 안에 있는 고객 정보를 모바일 앱에 전달한다.

7) 모바일 앱: 적립금액을 메인으로 고객 정보를 출력한다.

A.2 가맹점용 시스템

1) 사용자: POS 기기에 앱을 다운받는다.

2) 사용자: POS기 앱을 실행한다.

3) POS기 앱: “ID”, “Password”, “회원가입”을 입력하는 창을 출력한다.

4) 사용자: 계정 유무확인 또는 회원가입을 한다.

4.1) 계정이 있을 시 - ID와 Password를 입력하고 Join 버튼을 누른다.

4.2) 계정이 없을 시 - 가맹점의 계정정보를 입력하는 회원가입 화면을 출력한다.

4.2.1) POS기 앱: 가맹점의 계정 정보를 데이터베이스에 보낸다.

4.2.2) 시스템 동작: 해당 계정이 데이터베이스에 등록이 되는지 확인한다.

4.2.2.1) 등록 가능 - 해당 계정을 데이터베이스에 저장한다.

4.2.2.2) 등록 실패 - POS기 앱: “회원 가입 실패” 메시지를 출력한다.

5) POS기 앱: ID와 Password가 일치하는지 확인한다.

5.1) 일치 - POS기 앱: 해당 계정 정보를 데이터베이스에서 가져와 출력한다.

5.2) 불일치 - POS기 앱: “로그인 실패” 메시지를 출력한다.

6) 시스템 동작: 계정 안에 있는 가맹점 정보를 POS기 앱에 전달한다.

7) POS기 앱: 적립금액을 메인으로 가맹점 정보를 출력한다.

B 동전 관리

1) 사용자: 동전 적립 버튼을 선택한다.

2) 안드로이드 앱: 모바일 기기와 POS기가 NFC로 연결됨을 확인한다.

2.1) NFC가 연결되어 있는 경우 - 모바일 앱: 다음 단계로 이동한다.

2.2) NFC가 연결되어 있지 않은 경우- 모바일 앱: NFC가 연결되어있지 않다는 메시지를 출력한다.

3) 안드로이드 앱: 적립할 금액을 보여주는 초기화면을 출력한다.

4) 시스템 동작: 서버에 적립할 금액을 전송한 뒤 금액을 갱신한다.

5) 안드로이드 앱: 갱신된 금액을 화면에 출력한다.

C 환급

1) 사용자: 환급 버튼을 선택한다.

2) 시스템 동작: 서버에서 적립 금액과 계좌정보를 조회한다.

3) 시스템 동작: 기준환급금액 이상인지 확인한다.

3.1) 기준환급금액 이상일 경우 - 모바일 앱: 환급 금액과 계좌정보를 확인하는 창을 보여준다.

3.2) 기준환급금액 미만인 경우 - 모바일 앱: 기준환급금액 미달이라는 메시지를 출력한다.

4) 사용자: 환급을 원하는지 선택한다.

4.1) 환급을 원할 경우 - 모바일 앱: 고객정보에 있는 은행에 해당 금액과 계좌이체를 요청한다.

4.2) 환급을 원하지 않을 경우 - 모바일 앱: 초기화면으로 돌아간다.

5) 안드로이드 앱: 계좌이체 성공 시 성공적인 환급을 알리는 메시지를 출력한다.

12. Development Environment

12.1 Objective

이 장에서는 Piggy-B 시스템 개발 환경에 대해 서술한다.

12.2 Operating Systems

Figure 7 Windows 10

개발 환경 운영 체제로는 빠른 개발을 위해 기존 팀원들이 개발 환경을 구축하고 사용해 오던 Windows 10 을 선택하였다. Windows 10 은 마이크로소프트의 윈도우 계열의 개인용 컴퓨터 운영 체제로, 2015 년 7 월 15 일에 공개, 7 월 29 일에 출시 되었다. 이 외에도 리눅스, MAC OS 등 현재 많이 사용하고 있는 다른 운영체제에서도 개발 환경을 구축할 수 있도록 호환성에 초점을 맞추어 추후 유지 보수에 개발 환경 운영 체제에 대한 의존도를 제거하였다.

12.3 Running Enviornment

A Android

Figure 8 Android KitKat

스마트폰 운영체제로는 안드로이드 운영체제를 선택하였다. 애플 사의 iOS 는 개발 환경이 Mac OS X 로 한정되는 문제점과, 팀원들의 개발 경험을 고려했을 때 안드로이드가 더 적합하다고 판단되었다.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.3%
4.1.x	Jelly Bean	16	8.1%
4.2.x		17	11.0%
4.3		18	3.2%
4.4	KitKat	19	34.3%
5.0	Lollipop	21	16.9%
5.1		22	19.2%
6.0	Marshmallow	23	2.3%

Data collected during a 7-day period ending on March 7, 2016.
Any versions with less than 0.1% distribution are not shown.

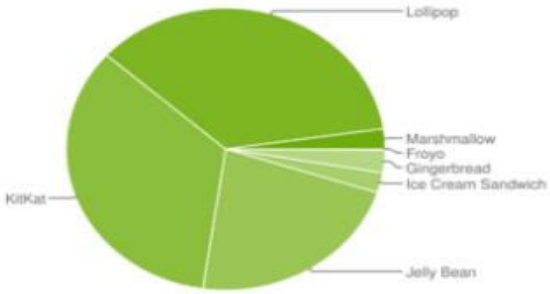

Figure 9 2016 Android 버전별 분포 Graph

지난 3 월 구글이 공개한 2016 년 3 월 안드로이드 플랫폼 분포도 현황에 따르면 롤리팝 보급률이 36.1%, 킷캣이 34.3%순으로 높았고, 가장 최신 OS 인 마시멜로우는 설치 비율이 2.3%에 그쳤다. 안드로이드는 버전 파편화 문제가 있는데, Piggy-B 시스템은 안드로이드 4.4 킷캣 이상 버전을 지원함으로써 전세계 기기의 약 73%에서 동작할 수 있도록 개발한다.

12.4 IDE

A Android studio 2.1.1

Figure 10 Android Studio

Android studio(안드로이드 스튜디오)는 안드로이드를 위한 통합 개발 환경이다. 2013 년 5 월 16 일, 구글 I/O 컨퍼런스에서 구글의 제품 관리자인 Ellie Powers 에 의해 발표되었다. 아파치 라이선스 2.0 으로 배포된다. 해당 시스템의 개발에서는 안드로이드 스튜디오 2.1.1 버전을 사용한다.

B MySQL

Figure 11 MySQL

MySQL 은 표준 데이터베이스 질의 언어인 SQL(Structured Query Language)을 사용하는 개방 소스의 관계형 데이터베이스 관리 시스템이다. 2016 년 기준 80% 이상의 시장 점유율을 차지하고 있으며 매우 빠르고 유연하며 사용하기 쉬운 특징이 있다. 다중 사용자, 다중 스레드를 지원하고, C, C++, Eiffel, 자바, 펄, PHP, Python 스크립트 등을 위한 응용 프로그램 인터페이스(API)를 제공한다. 유닉스나 리눅스, 윈도 운영 체제 등에서 사용할 수 있다. LAMP, 즉 리눅스 운영 체제와 Apache 서버 프로그램, MySQL, PHP 스크립트 언어 구성은 상호 연동이 잘되면서도 오픈 소스로 개발되는 무료 프로그램이어서 홈 페이지나 쇼핑몰 등 일반적인 웹 개발에 널리 이용되고 있다.

1995 년 5 월 스웨덴의 MySQL AB 사에 의해 최초의 MySQL 버전이 출시되었으며, 1998 년 윈도우 용 첫 버전이 출시되었다. 2008 년 썬 마이크로시스템즈(Sun Microsystems)가 MySQL 를 인수한 후 버전 5.1 을 출시하였으며, 2010 년 6 월 오라클이 썬 마이크로시스템즈를 통합 인수한 후, 같은 해 12 월 버전 5.5 를 출시하였다. 2015 년 10 월 버전 5.7 이 출시되었다.

12.5 Programming Language

A JAVA 8

Figure 12 JAVA

자바는 썬 마이크로시스템즈의 제임스 고슬링(James Gosling)과 다른 연구원들이 개발한 객체 지향적 프로그래밍 언어이다. 썬 마이크로시스템즈가 오라클에 인수된 후 오라클에서 무료로 제공하고 있다. 1991 년 그린 프로젝트(Green Project)라는 이름으로 시작했으며 1995 년에 발표되었다. 처음에는 가전제품 내에 탑재해 동작하는 프로그램을 위해 개발했지만 현재 웹 어플리케 이션 개발에 가장 많이 사용하는 언어 가운데 하나이고, 모바일 기기용 소프트웨어 개발에도 널리 사용하고 있다. 현재 버전 9 까지 출시했다.

자바의 개발자들은 유닉스 기반의 배경을 가지고 있었기 때문에 문법적인 특성은 파스칼이 아닌 C++의 조상인 C 언어와 비슷하다. 자바를 다른 컴파일언어와 구분 짓는 가장 큰 특징은 컴파일 된 코드가 플랫폼 독립적이라는 점이다. 자바 컴파일러는 자바 언어로 작성된 프로그램을 바이트코드라는 특수한 바이너리 형태로 변환한다. 바이트코드를 실행하기 위해서는 JVM(자 바 가상 머신, Java Virtual Machine)이라는 특수한 가상 머신이 필요한데, 이 가상 머신은 자 바 바이트코드를 어느 플랫폼에서나 동일한 형태로 실행시킨다. 때문에 자바로 개발된 프로그램은 CPU 나 운영 체제의

종류에 관계없이 JVM 을 설치할 수 있는 시스템에서는 어디서나 실 행할 수 있으며, 이 점이 웹 애플리케이션의 특성과 맞아떨어져 폭발적인 인기를 끌게 되었다. 또한 안드로이드 앱 개발 언어로 자바가 선택되었다.

B PHP

Figure 13 PHP

PHP(Hypertext Preprocessor)는 프로그래밍 언어의 일종이다. 1995 년 라스무스 러도프가 처음 만든 것으로, 2015 년 12 월 3 일 공개된 Php7.0.0 까지 존재한다. 원래는 동적 웹 페이지를 만들기 위해 설계 되었으며, 이를 구현하기 위해 PHP 로 작성된 코드를 HTML 소스 문서 안에 넣으면 PHP 처리 기능이 있는 웹 서버에서 해당 코드를 인식하여 작성자가 원하는 웹 페이지를 생성한다. 많은 서버 측 오픈 소스 소프트웨어는 PHP 로 구현되었다. PHP 는 텍스트, 특히 HTML 의 처리에 강점을 가지고 있다. URL 의 파싱이나 폼 처리, 정규 표현식 등이 그 한 예이다. 또한 다양한 데이터베이스를 지원하므로 데이터베이스와 사용자간의 다리 역할도 잘 수행한다.

12.6 Version Control

A GitHub

Figure 14 GitHub

GitHub 는 분산 버전 관리 툴인 깃(Git)을 사용하는 프로젝트를 지원하는 웹 호스팅 서비스이다. 깃(Git)은 2005 년에 개발된 분산형 버전 관리 시스템으로, 오픈 소스 소프트웨어이며 리눅스를 만든 리누스 토발즈와 주니오 하마노가 개발했다. 깃을 이용하면 누가 어떤 코드를 수정했는지 기록하고 추적할 수 있다. 많은 사람들이 함께 소프트웨어를 개발할 때 유용하며, 관리자는 여러 사람의 코드 중 일부를 합쳐가며 완성본을 만들어갈 수 있다. GitHub 는 웹 그래픽 기반으로 깃을 이용할 수 있게 만들었기 때문에 보다 편하게 이용할 수 있는 서비스다. GitHub 는 가장 인기있는 Git 호스팅 사이트이며, 인기있는 오픈 소스 코드 저장소로 꼽힌다. GitHub 는 오픈소스 소프트웨어의 중심지 역할을 하면서 오픈 소스 프로젝트가 널리 퍼지는 데 크게 기여하고 있다.

13. Index

13.1 Figure Index

13.2 Table Index

13.3 Diagram Index

14. Reference

Gliffy	http://www.wiu.edu/coehs/techinsights/blog/?p=635
Windows 10	http://www.microsoftstore.com/store/mskr/ko_KR/cat/categoryID.66834071?tduid=(2d6455d16b7f5e285cd2e2be8300cd15)(225151)(2223360)(1-3643-__-KyPDlcOVKQ==)()
Android KitKat	http://comterman.tistory.com/971
안드로이드 버전 분포율	http://www.etnews.com/20160310000008
JAVA	https://blog.newrelic.com/2014/12/08/10-ways-java-money/
깃허브	https://digitalfellows.commons.gc.cuny.edu/2015/03/10/intro-to-github-part-i/
안드로이드 스튜디오	http://www.phonearena.com/news/How-to-install-Android-SDK-via-Android-Studio-on-Windows_id69854
MySQL	https://en.wikipedia.org/wiki/File:MySQL.svg http://terms.naver.com/entry.nhn?docId=3353294&cid=40942&categoryId=32840
PHP	http://idchowto.com/?p=1840 https://ko.wikipedia.org/wiki/PHP